UVA CS 4774: Machine Learning

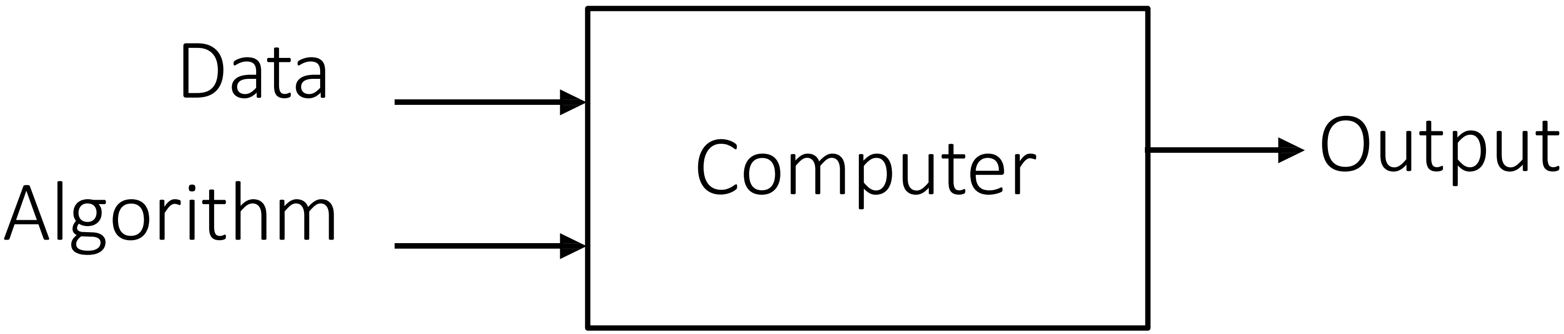
S6: Lecture 29: Course Review Sessions

Dr. Yanjun Qi

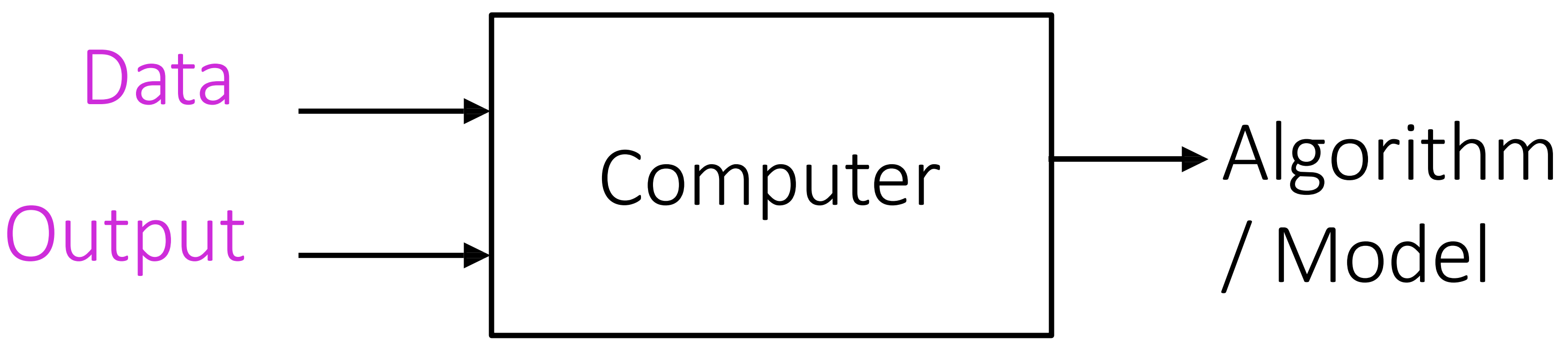
University of Virginia
Department Of Computer Science

09/02

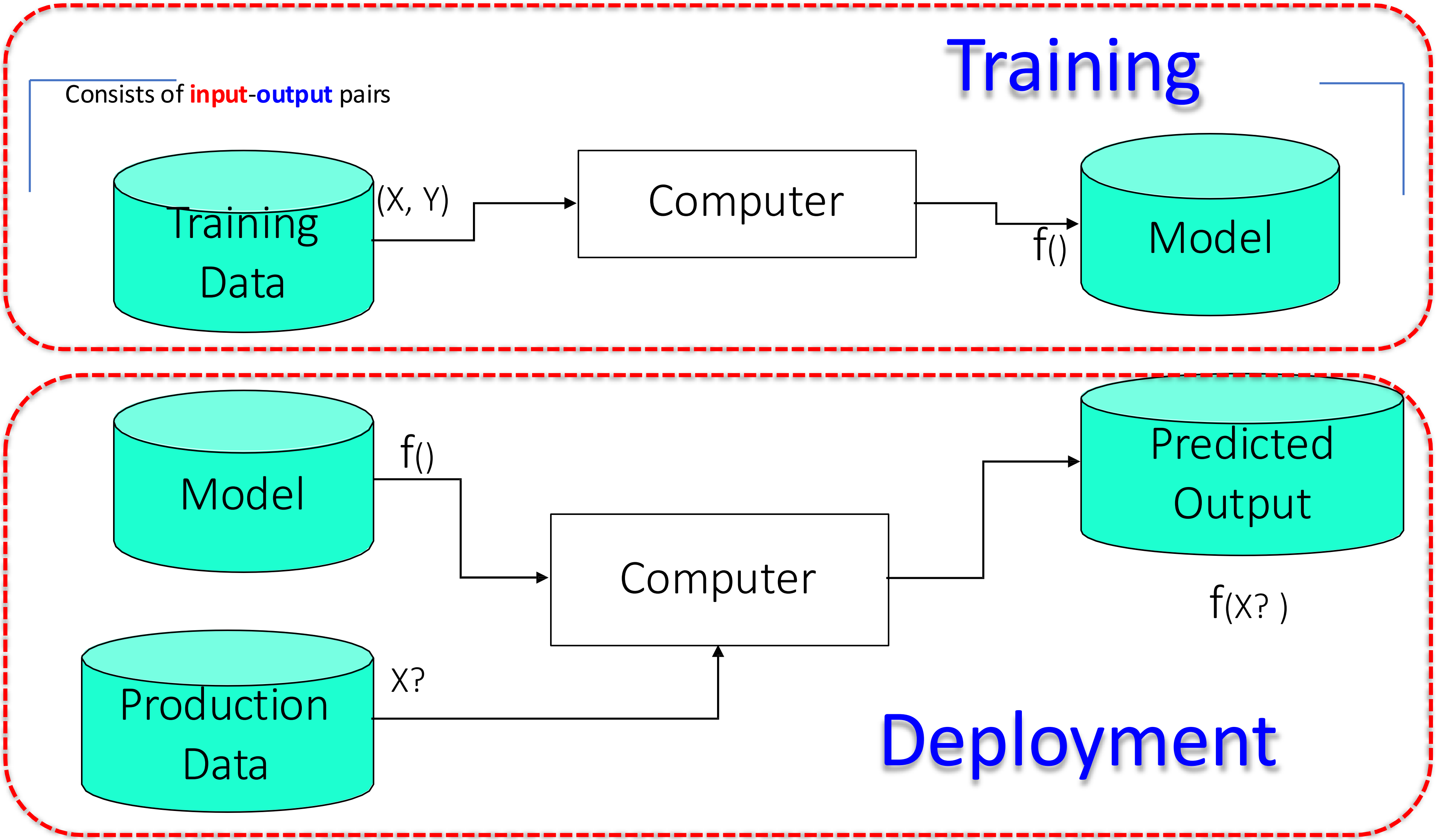
Traditional Programming



Machine Learning



Two Modes of Machine Learning



Machine Learning in a Nutshell

Data

Task

Representation

Score Function

Search/Optimization

Models, Parameters

Hardware

ML grew out of work in AI

Optimize a performance criterion using example data or past experience,

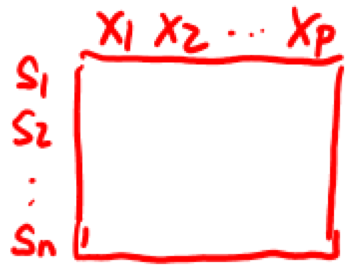
Aiming to generalize to unseen data

Rough Sectioning of this Course

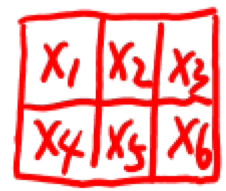
- S1. Basic Supervised Regression + Tabular Data
- S2. Basic Deep Learning + 2D Imaging Data
- S3. Generative and Deep + 1D Sequence Text Data
- S4. Advanced Supervised learning + Tabular Data
- S5. Not Supervised
- S6: Wrap Up + (a few invited tasks, e.g. on AWS)

Course Content Plan ➔ Regarding Data

- ❑ Tabular / Matrix



- ❑ 2D Grid Structured: Imaging

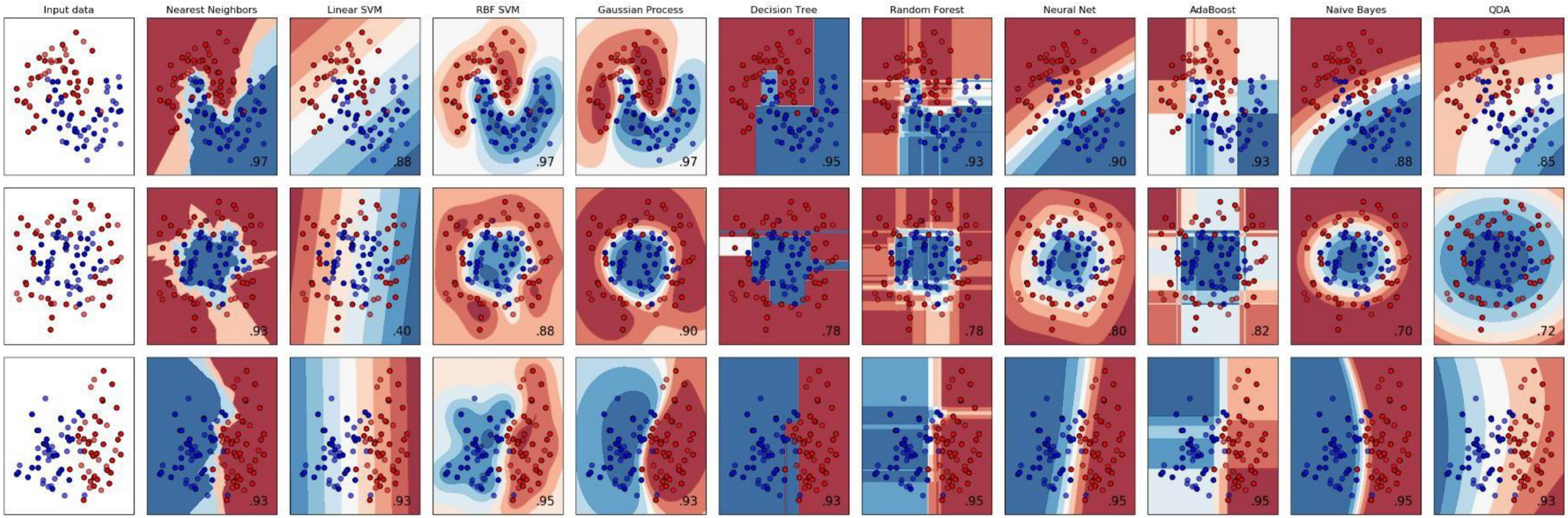


- ❑ 1D Sequential Structured: Text
- ❑ Graph Structured (Relational)
- ❑ Set Structured / 3D /

Course Content Plan → Regarding Tasks

- ❑ Regression (supervised) ← Y is a continuous
- ❑ Learning theory ← About f()
- ❑ Classification (supervised) ← Y is a discrete
- ❑ Unsupervised models ← NO Y
- ❑ Graphical models ← About interactions among Y,X1,. Xp
- ❑ Reinforcement Learning ← Learn to Interact with environment

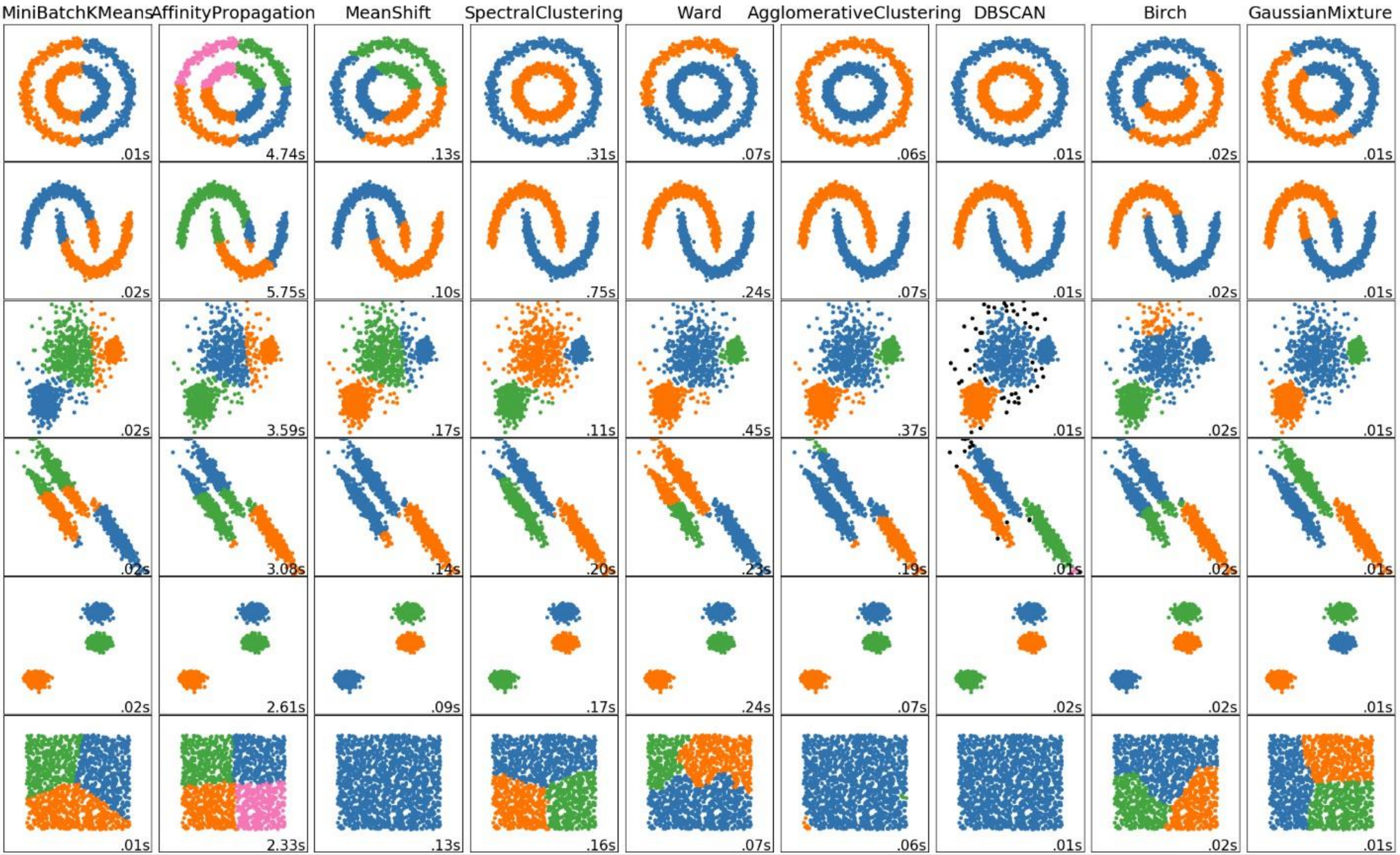
https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html



- ✓ different assumptions on data
- ✓ different scalability profiles at training time
- ✓ different latencies at prediction (test) time
- ✓ different model sizes (embedability in mobile devices)
- ✓ different level of model interpretability / robustness

Adapt from Olivier Grisel's talk

https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html



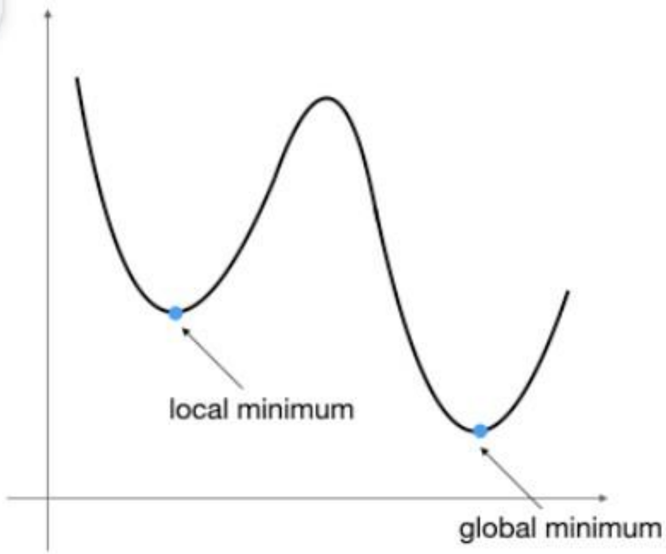
- ✓ different assumptions on data
- ✓ different scalability profiles
- ✓ different model sizes (embedability in mobile devices)

09/09

Quiz 3 Plus

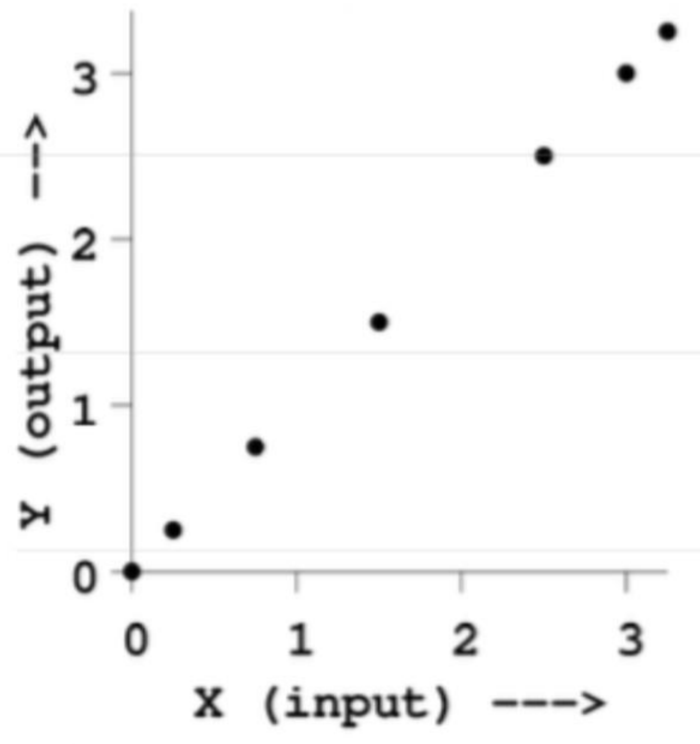
2. True or False? Gradient descent always finds the global minimum. (Hint: Imagine the initial value starts from local minimum, the gradient there is 0)

Multiple choice



- False
- True

Question 3.1. Linear Regression+ Train-Test Split



Quiz 3 Plus

Figure 1: A reference dataset for regression with one real-valued input (x as horizontal axis) and one real-valued output (y as vertical axis).

What is the mean squared training error when running linear regression to fit the data ? (i.e., the model is $y = \beta_0 + \beta_1 x$). Assuming the rightmost three points are in the test set, and the others are in the training set. (you can eyeball the answers.)

L 3,4- Reading Questions

1. Fundamentals of Linear Regression

- What does it mean for a dataset to be a good fit for linear regression?
- Does linear supervised regression only work with data that is already somewhat linear?
- When is it a good time to use linear regression, and under what conditions will it perform best?
- Where is linear regression used in real-world applications today?
- What challenges exist in making linear regression models robust and trustworthy?
- How should we interpret regression coefficients when features are correlated? Does GD handle multicollinearity?
- What does the “bias” term represent conceptually?

L 3,4- Reading Questions

- 2. Loss / Cost Functions
- What exactly is the meaning of Mean Squared Error (MSE), Mean Absolute Error (MAE), Sum of Squared Errors (SSE)?
- When is MSE preferred over MAE, and what are the trade-offs?
- Why is SSE chosen in linear regression instead of MAE?
- What is the purpose of the ½ factor in quadratic loss?
- How do loss functions differ for convex, concave, and saddle point graphs?
- Where did the SSE loss measurement originate from?
- What is the difference between objective, cost, and loss function terminology?
- How do we choose performance metrics (MSE, MAE, R^2, others) and when should multiple metrics be combined?

L 3,4- Reading Questions

- 3. Gradient Descent & Optimization
- How does gradient descent (GD) work conceptually?
- What's the difference between GD, stochastic GD (SGD), and mini-batch GD?
- How do we choose learning rate (α) values? Are they fixed or dynamic?
- How do we pick good starting points for GD?
- How does GD behave near local minima, saddle points, or flat regions?
- Are there ways for SGD to escape local minima/saddle points?
- What are good batch sizes, and how do they affect convergence?
- What are the limitations of GD and strategies to overcome them?
- How do we evaluate convergence and know when to stop?
- Could you show a full worked-out example of optimizing with GD step by step?

L 3,4- Reading Questions

- 4. Normal Equation vs Iterative Methods
- When should we use the Normal Equation versus Gradient Descent or SGD?
- What are the computational trade-offs between closed-form (Normal Eq.) and iterative (GD/SGD) methods?
- What happens if the feature matrix X does not have full rank?
- Why is Strassen’s algorithm for matrix multiplication not always the default, despite being faster in theory?

- 5. Model Selection & Trade-offs
- How do we know when to choose linear regression vs. more complex models (e.g., Random Forest, SVC)?
- How do we evaluate trade-offs between generalization, efficiency, scalability, and interpretability?
- How does context influence model selection and visualization choices?
- How are these classical regression/optimization topics applied to modern LLMs like ChatGPT or Alexa?

L 3,4- Reading Questions

- 6. Training, Testing, and Generalization
- How do we decide the split between training and testing data? (e.g., 80/20 rule)
- Is there an optimal ratio of training to test size?
- What does it mean for a model to generalize well? Does it just mean low error?
- Why is performance on training data not a good indicator of generalization?
- What happens if train/test sets have slightly different distributions (distribution shift)?
- When should validation sets be introduced in addition to train/test splits?
- How does generalization relate to overfitting/underfitting (residual patterns, feature poisoning examples)?

- 7. Matrix & Representation Issues
- Why represent regression in matrix form? How does it help computation and parallelization?
- What's the difference between summation form and matrix form of the loss function?
- How do row vs. column vectors work in NumPy?
- What does it mean for a matrix to be full rank, and why does it matter?

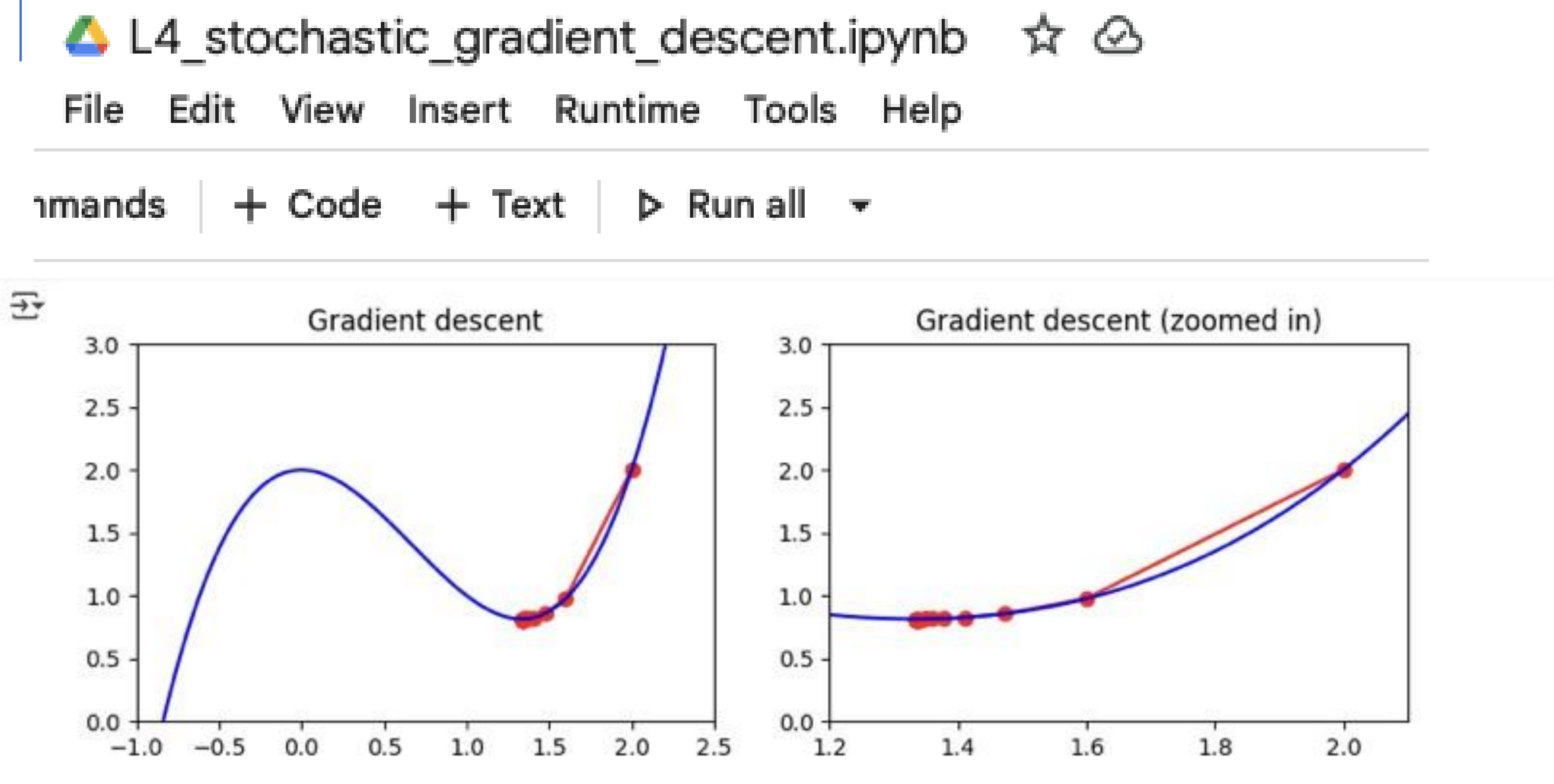
Matrix Representation (p53-)

Lecture 3: Linear Regression Basics

Many architecture details and Algorithm details to consider

- (1): Data parallelization through CPU SIMD / Multithreading/ GPU parallelization / ....
- (2): Memory hierarchical / locality
- (3): Better algorithms, like Strassen's Matrix Multiply and many others

Learning Rate Code Run



09/16-23

Quiz 3 Plus

1. True or False: For linear regression, the loss function (sum of squared errors) is convex, meaning gradient descent is guaranteed to find the global minimum if the learning rate is chosen appropriately.

- (●) True
- ( ) False

Quiz 3 Plus

2. In linear regression, what is the role of the intercept term?

- ( ) It scales the input features
- (•) It shifts the regression line vertically
- ( ) It reduces the variance of predictions
- ( ) It normalizes the input data

Quiz 3 Plus

4. Suppose we want to minimize the function f(w)=w^2+4w using gradient descent. The initial value is w_0=2, and the learning rate is α=0.1. What will be the value of w_1 after one gradient descent update?

- w_1 = 1.6
- w_1 = 2.4
- w_1 = −2
- w_1 = 1.2

Add answer feedback

Quiz 3 Plus

3. Given the model and loss function, which of the following will be iteratively optimized to minimize the loss by gradient descent?

Multiple choice

- ○ Input and output
- ○ Model type (e.g. linear model or nonlinear model)
- ○ Model parameters ✓
- ○ Add option or add "Other"

L5 – e.g. LR with radial-basis functions

- E.g.: LR with RBF regression:

$$\hat{y} = \theta_0 + \sum_{j=1}^{m} \theta_j \varphi_j(x) = \varphi(x)^T \theta$$

$$\varphi(x) := \left(1, K_{\lambda_1}(x, r_1), K_{\lambda_2}(x, r_2), K_{\lambda_3}(x, r_3), K_{\lambda_4}(x, r_4)\right)^T$$

$$\theta^* = \left(\varphi^T \varphi\right)^{-1} \varphi^T \vec{y}$$

bias

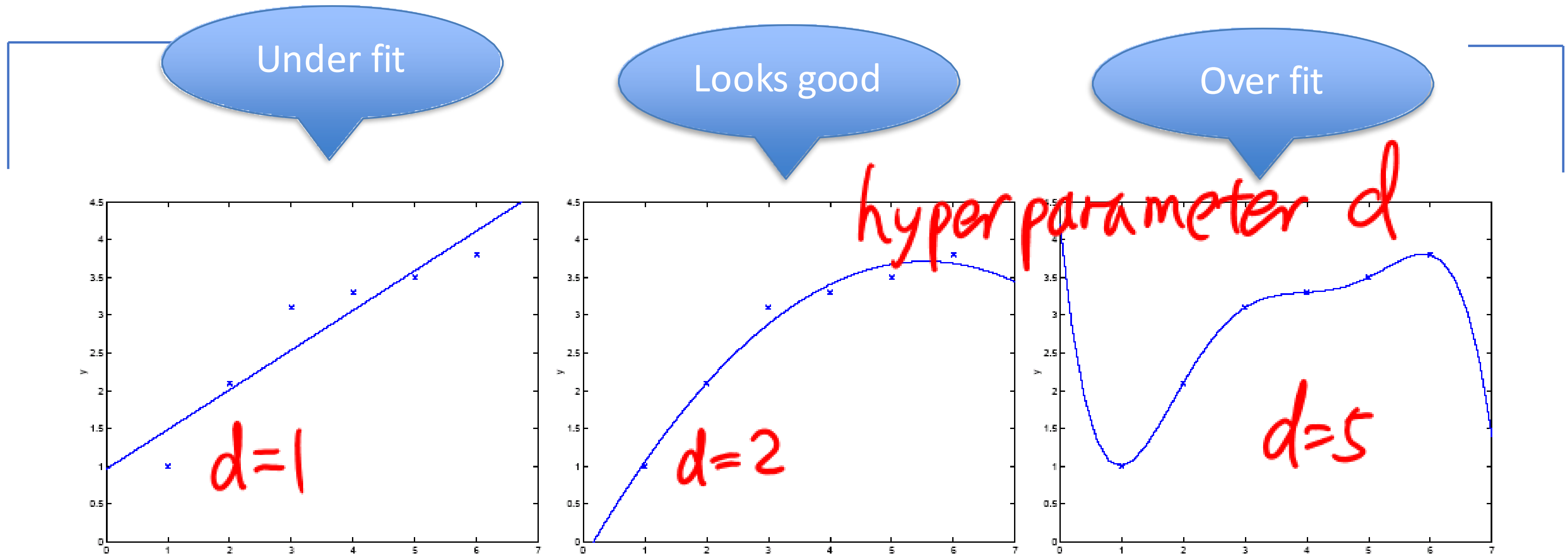
$$\vec{\theta} = [\theta_0, \overbrace{\theta_1, \theta_2, \theta_3, \theta_4}^{\text{weights}}]^T$$

$$\left.\begin{array}{cccc} r_1 & r_2 & \lambda_3 & r_4 \\ \lambda_1 & \lambda_2 & r_3 & \lambda_4 \end{array}\right\} \text{hyper para}$$

L6: Main issues: Model Selection

- How to select the right model type? How to select hyperparameter for a model type?
  - E.g. what polynomial degree d for polynomial regression
  - E.g., where to put the centers for the RBF kernels? How wide?
  - E.g. which basis type? Polynomial or RBF?

What Model Order to Select?

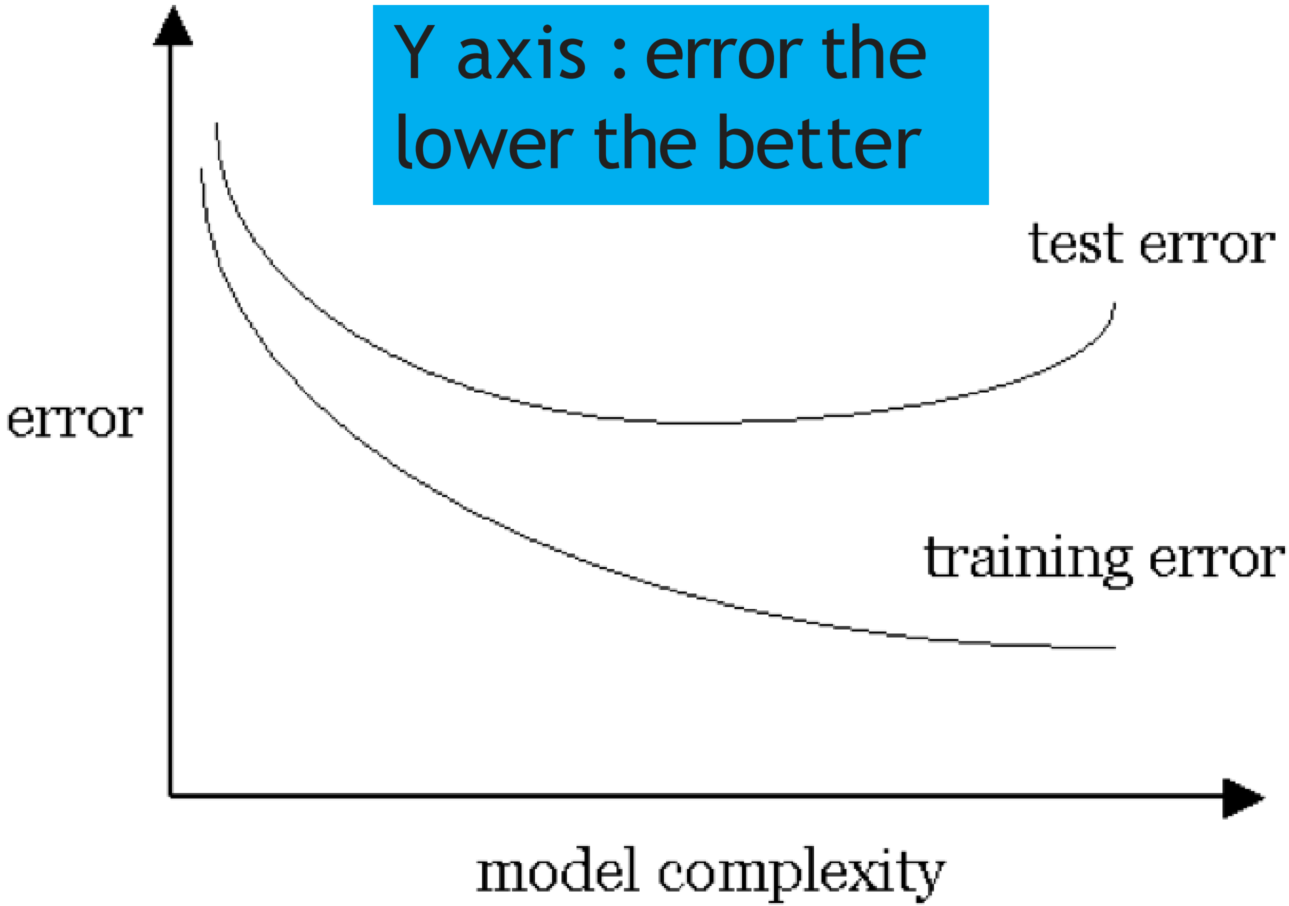


$$y = \theta_0 + \theta_1 x \qquad y = \theta_0 + \theta_1 x + \theta_2 x^2 \qquad y = \sum_{j=0}^{5} \theta_j x^j$$

Generalisation: learn function / hypothesis from past data in order to "explain", "predict", "model" or "control" new data examples

(a) Train-validation /
(b) K-fold Cross Validation /

A Plot for Model Selection



k-CV on train to choose model and hyperparameter / then a separate test set to assess future performance

Polynomial Regression Code Run

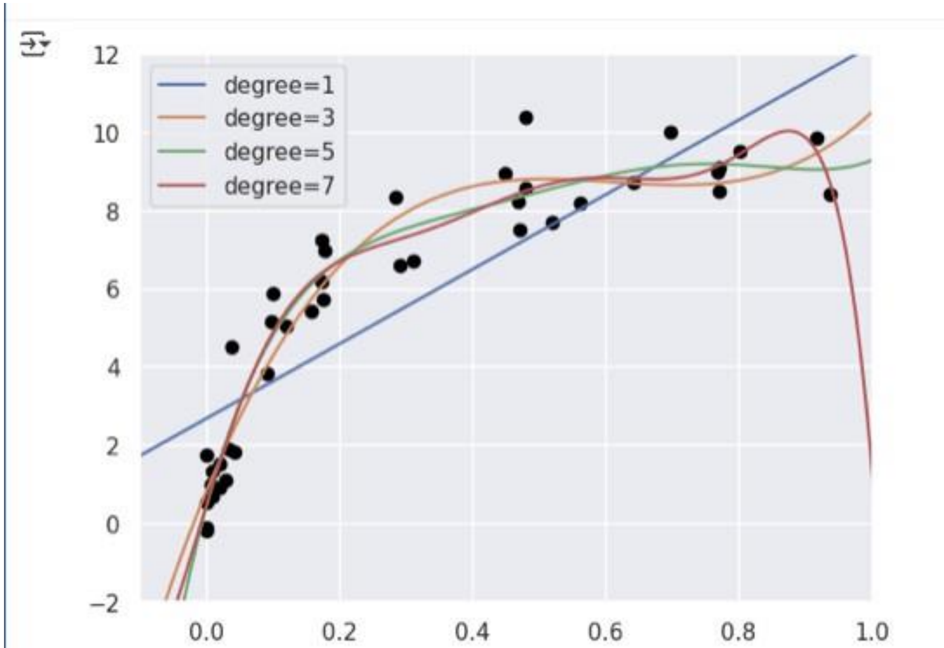
L5-Poly-Regression.ipynb

File Edit View Insert Runtime Tools Help

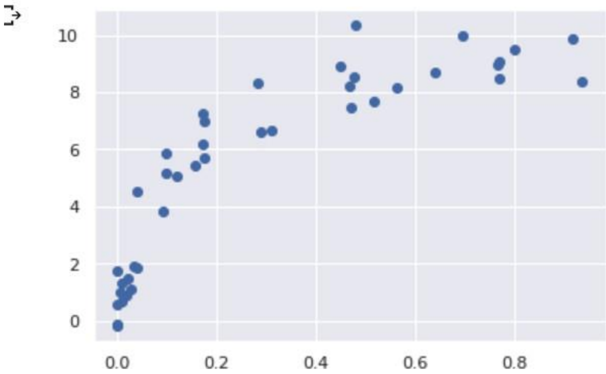
ımands + Code + Text ▷ Run all

More Regression / Modified from :

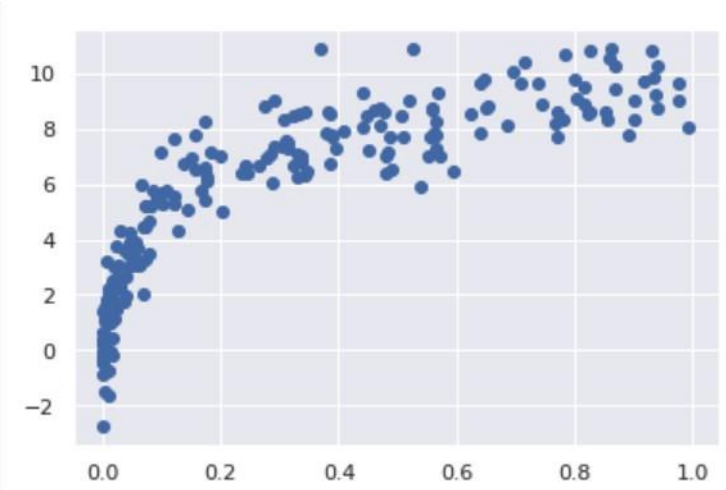
1. https://github.com/jakevdp/PythonDataScienceHandbook
2. https://jakevdp.github.io/PythonDataScienceHandbook/05.03-hyperparameters-and-model-validation.html



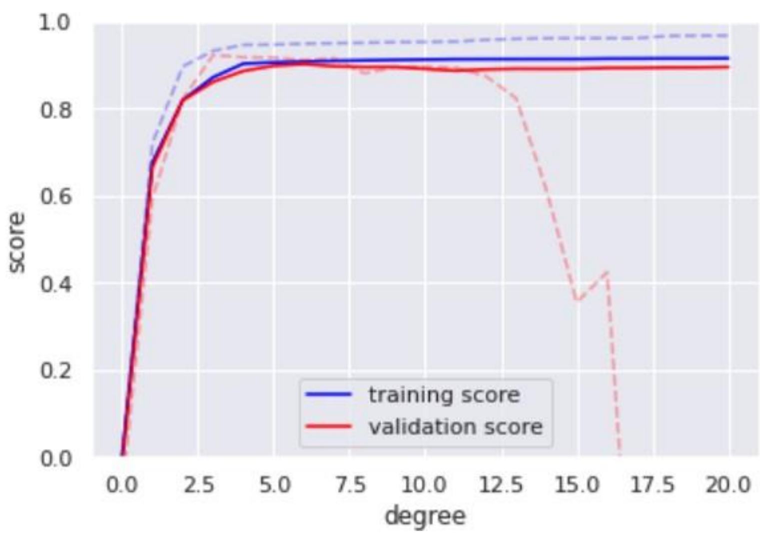
```
X, y = make_data(40)
plt.scatter(X, y);
```



```
X2, y2 = make_data(200)
plt.scatter(X2.ravel(), y2);
```



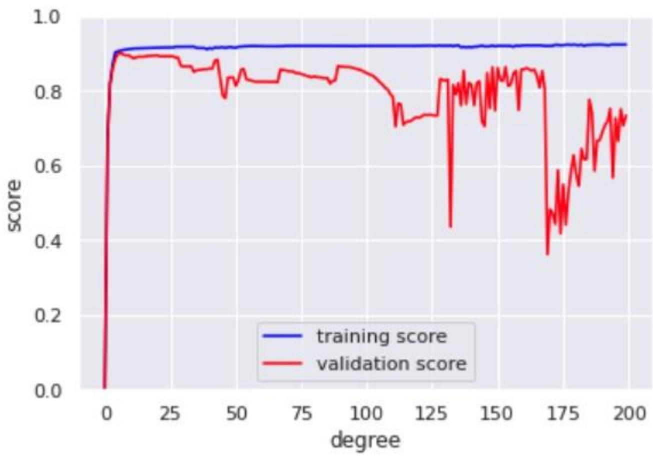
```
plt.plot(degree, np.median(train_score2, 1), color='blue'
plt.plot(degree, np.median(val_score2, 1), color='red', l
plt.plot(degree, np.median(train_score, 1), color='blue',
plt.plot(degree, np.median(val_score, 1), color='red', al
plt.legend(loc='lower center')
plt.ylim(0, 1)
plt.xlabel('degree')
plt.ylabel('score');
```



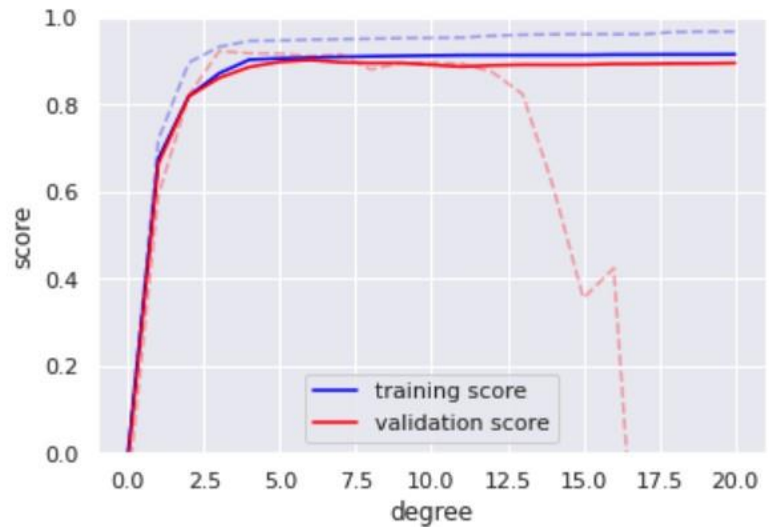
Behavior of the validation curve:

- the model complexity
- the number of training points

```python
X2, y2 = make_data(200)

degree = np.arange(200)
train_score2, val_score2 = validation_curve(PolynomialReg
                                            'polynomialfe

plt.plot(degree, np.median(train_score2, 1), color='blue'
plt.plot(degree, np.median(val_score2, 1), color='red', l
plt.legend(loc='lower center')
plt.ylim(0, 1)
plt.xlabel('degree')
plt.ylabel('score');
```




```python
X2, y2 = make_data(200)
degree = np.arange(21)
train_score2, val_score2 = validation_curve(PolynomialReg
                                            'polynomialfe

plt.plot(degree, np.median(train_score2, 1), color='blue'
plt.plot(degree, np.median(val_score2, 1), color='red', l
plt.plot(degree, np.median(train_score, 1), color='blue',
plt.plot(degree, np.median(val_score, 1), color='red', al
plt.legend(loc='lower center')
plt.ylim(0, 1)
plt.xlabel('degree')
plt.ylabel('score');
```



Interesting Relation between

- the right range of model complexity
- the number of training points

Quiz 4 plus

Which of the following statements about Leave-One-Out Cross Validation (LOOCV) is true?

Multiple choice

- It wastes a large fraction of training data.
- It has low variance but high computational cost. ✓
- It provides the same estimate as a large k-fold CV with k close to 1.
- It is cheaper than k-fold cross validation.
- Add option or add "Other"

Answer key (2 points)

Required

need to make assumptions that are able to generalize

- **Underfitting:** model is too "simple" to represent all the relevant characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too "complex" and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

A Gentle Touch of Bias - Variance Tradeoff

(More details … Later)

Slide credit: L. Lazebnik

09/30

09/30/2025 Assignments

- HW2 is due this coming Sunday midnight!
  - Using HW1 code pieces as components;
  - If you struggle with HW1, please contact TA @Haochen ASAP

- HW1 grading is work-in-progress,
  - Grades will be released by next Tuesday class time
  - We posted the guide from TA in Canvas

- Course vote:
  - New Survey that needs your vote on
    - 1. back to lecture in-person twice a week?
    - 2. If not, best way to use the in-person session:
      - Quiz to continue
      - + Project discussions
        - Interested in Shark Tank alike setup? idea screening, pitch talk, demo ...

Project Process

- Format:
  - Team, individual …
  - Shark Tank alike Screening ? – https://en.wikipedia.org/wiki/Shark_Tank
  - Next week – Idea collection

- Final deliverables:
  - (1) Code (Github PR to course project repo)
  - (2) Poster presentation class wide (Date: TBD)
  - (3) Video Demo (TBD)

09/30/2025 Roadmap

- TA to go over HW1
- One UVA ML club to introduce their setups and projects
- Q5
- Review Q4
- Review QA for L5-L7

L7: Regularized multivariate linear regression

Data: X

Task: y

Representation: : x, f()

Score Function: L()

Search/Optimization : argmin()

Models, Parameters

X: Tabular

Regression: y continuous

Y = Weighted linear sum of Xs

Sum of Squared Error (Least Squared) + Regularization

Revised Normal Equation / revised GD / SGD

Regression weights and bias => Regularized: Robust / Interpretable

L7: Regularized multivariate linear regression

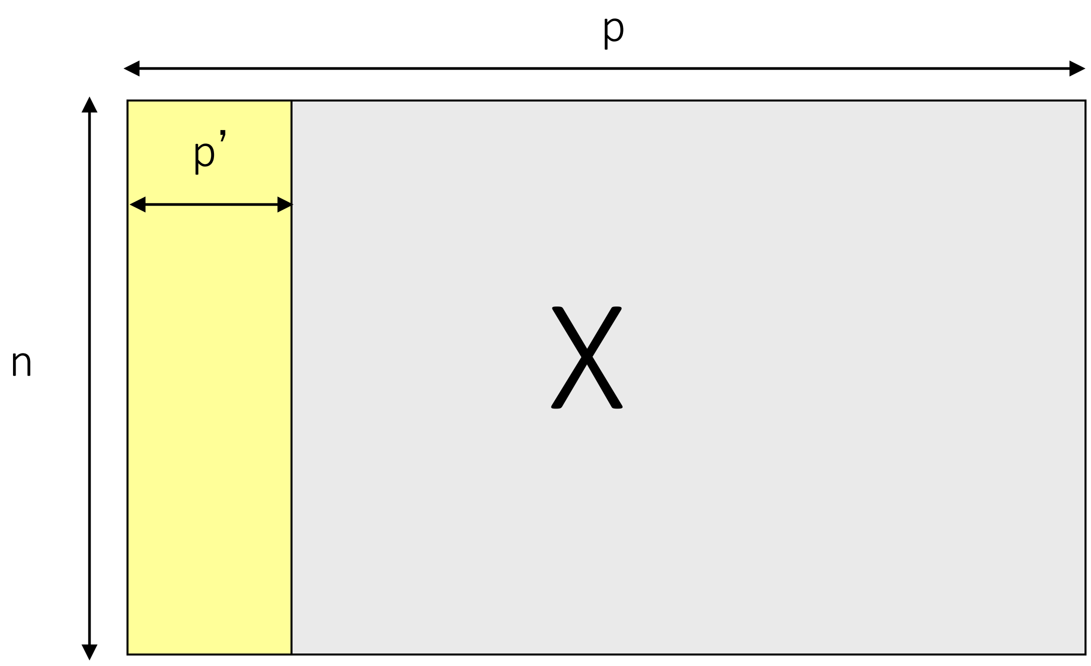
We aim to make our trained model

- 1. Generalize Well

- 2. Computational Scalable and Efficient

- 3. Trustworthy: Robust / Interpretable
  - Especially for some domains, this is about trust!

Large p, small n: How? $p \to p' \Rightarrow f$ easy to understand



Regularized multivariate linear regression

- Model: $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p$
  - LR estimation: $\arg\min \sum \left(Y - \hat{Y} \right)^2$
  - LASSO estimation: $\arg\min \sum_{i=1}^{n} \left(Y - \hat{Y} \right)^2 + \lambda \sum_{j=1}^{p} \left| \beta_j \right|$
  - Ridge regression estimation: $\arg\min \sum_{i=1}^{n} \left(Y - \hat{Y} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$

42/54

Error on data + Regularization

Ridge Regression / L2 Regularized Regression

$$\beta^* = \left(X^T X\right)^{-1} X^T \vec{y}$$

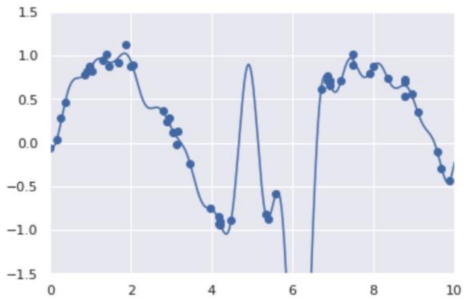
- If not invertible, a classical solution is to add a small positive element to diagonal

$$\beta^* = \left(X^T X + \lambda I\right)^{-1} X^T \vec{y}$$

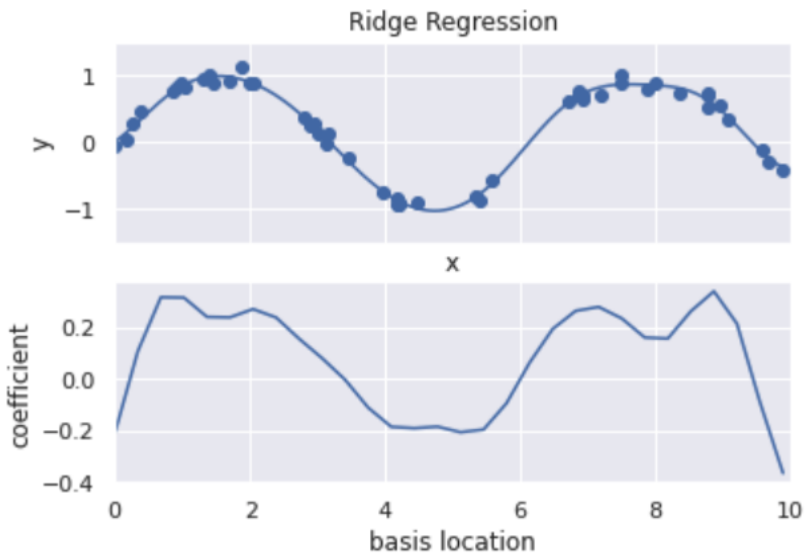
By convention, the bias/intercept term is typically not regularized.
Here we assume data has been centered … therefore no bias term

Overfitting: Can be Handled by Regularization

A regularizer is an additional criteria to the loss function to make sure that we don't overfit. It's called a regularizer since it tries to keep the parameters more normal/regular




```python
from sklearn.linear_model import Ridge
model = make_pipeline(GaussianFeatures(30), Ridge(alpha=0.1))
basis_plot(model, title='Ridge Regression')
```



WHY and How to Select λ?

- 1. Generalization ability
  ➔ k-folds CV to decide

- 2. Control the bias and Variance of the model (details in future lectures)

L2: Squared weights penalizes large values more

L1: Sum of weights will penalize small values more

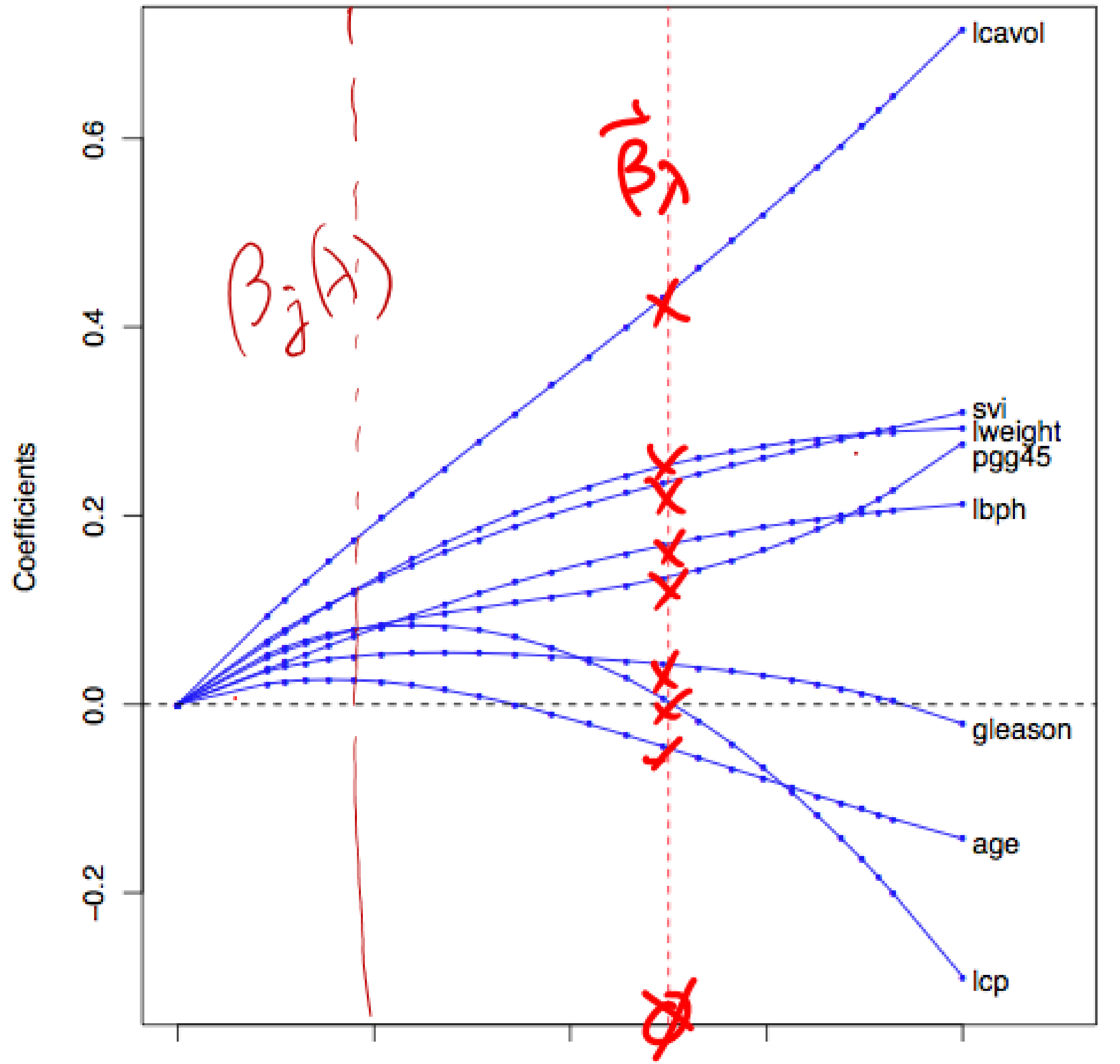
$$\sum_j \beta_j^2$$

$$\sum_j \left|\beta_j\right|$$

Regularization path of a Ridge Regression

when $X'X = I$ $\Rightarrow \frac{1}{1+\lambda}\beta_{OLS}$

Weight Decay



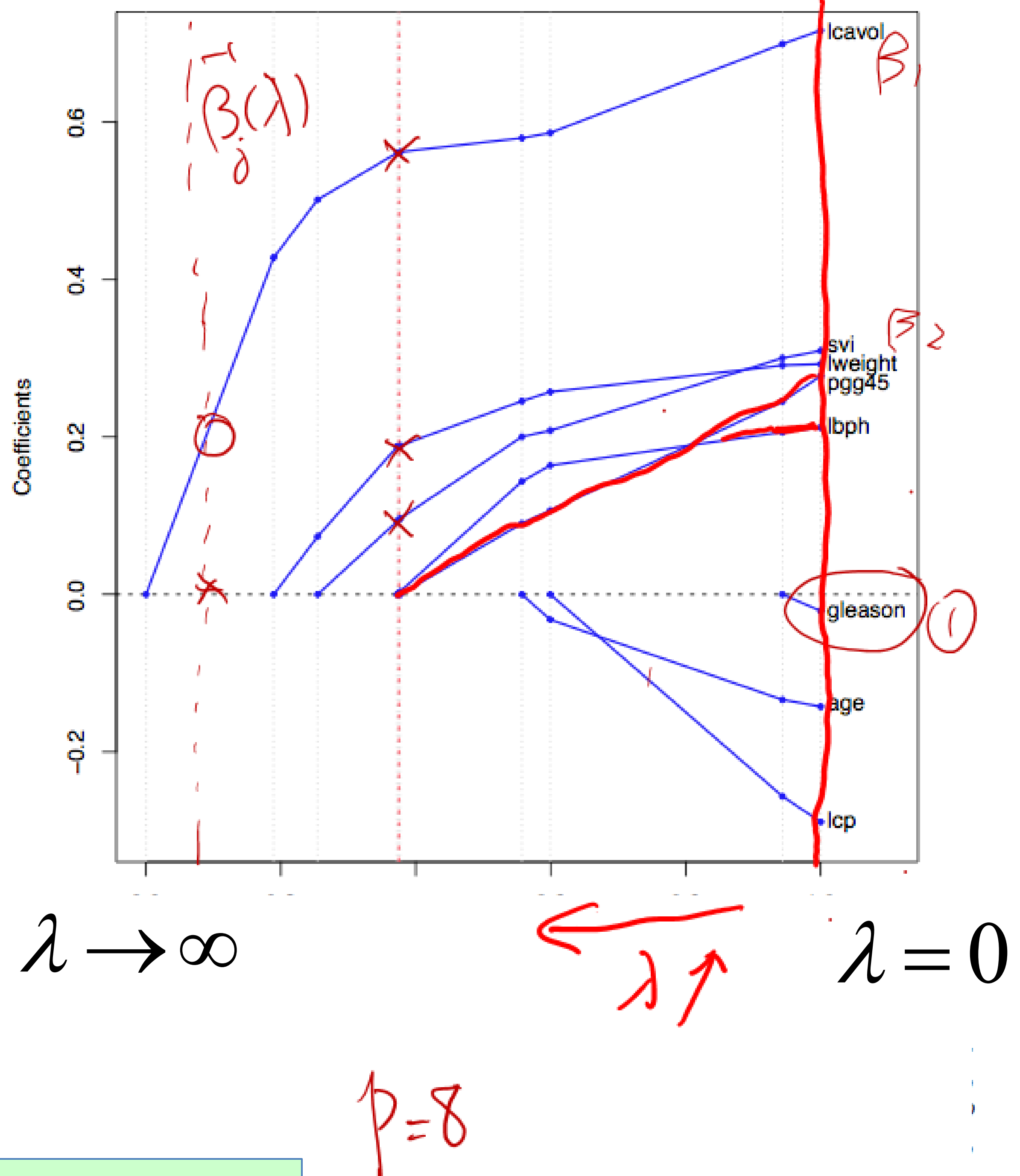
WHY and How to Select λ?

$\lambda \to \infty$ $\qquad$ $\lambda = 0$

An example with 8 features

Regularization path of a Lasso Regression

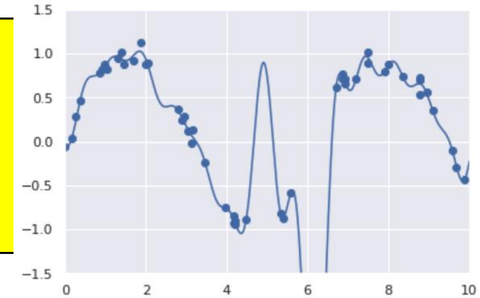
when varying λ, how β_j varies.



An example with 8 features

Overfitting: Can be Handled by Regularization

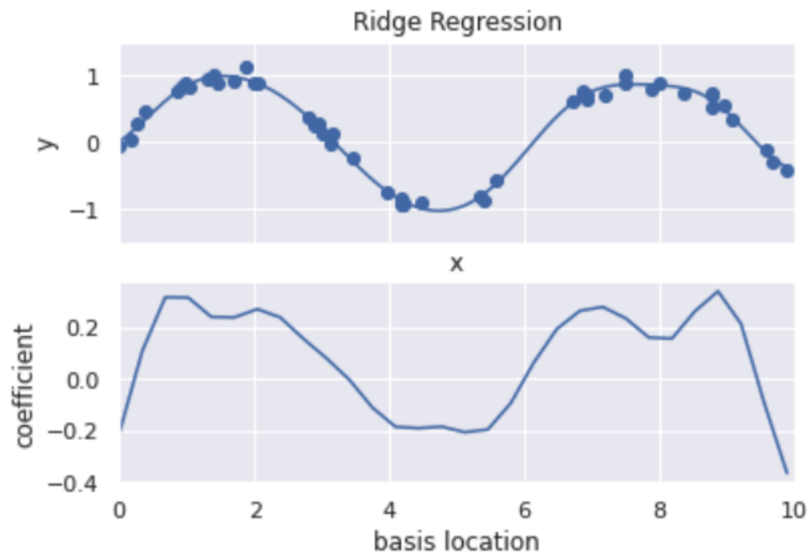
A regularizer is an additional criteria to the loss function to make sure that we don't overfit. It's called a regularizer since it tries to keep the parameters more normal/regular



code-run:

https://github.com/qiyanjun/2025Fall-UVA-CS-MachineLearningDeep/blob/main/notebook/L7_regularizedRegression_06_Linear_Regression.ipynb

```
from sklearn.linear_model import Ridge
model = make_pipeline(GaussianFeatures(30), Ridge(alpha=0.1))
basis_plot(model, title='Ridge Regression')
```



(Extra) Lasso (least absolute shrinkage and selection operator) / Squared Loss+L1

- The lasso is a shrinkage method like ridge, but acts in a nonlinear manner on the outcome y.
- The lasso is defined by

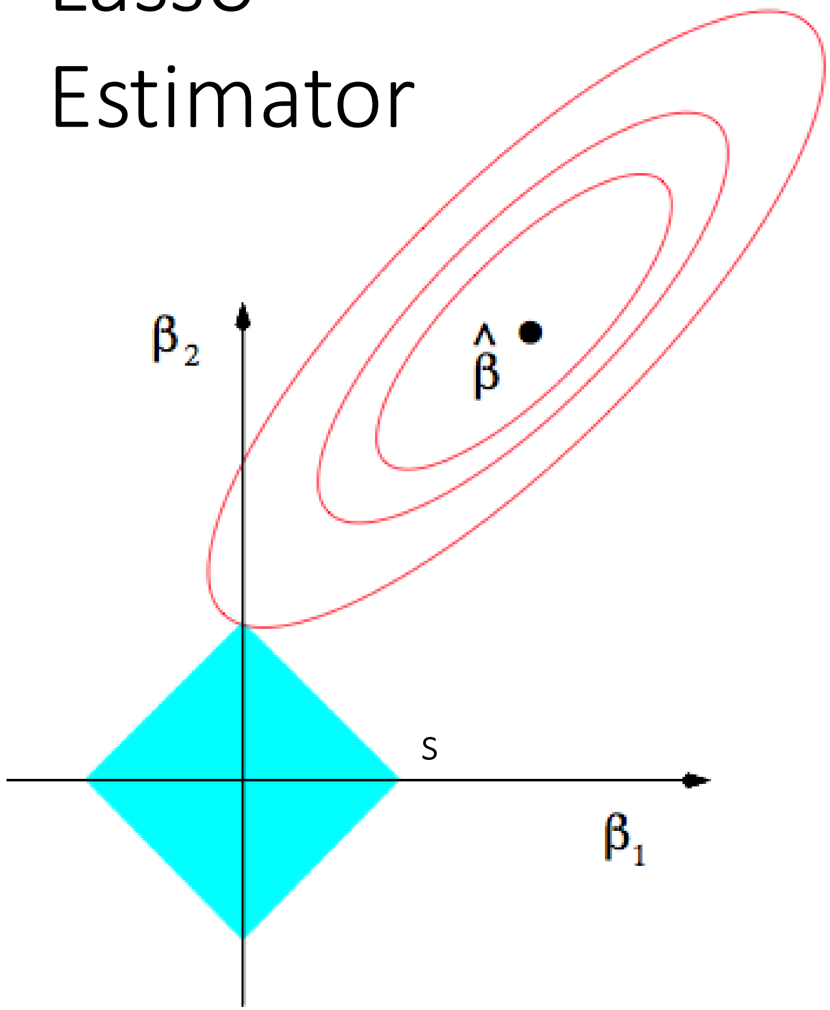
$$\sum_{i=1}^{n}(y_i - x_i^T\beta)^2$$

$$\hat{\beta}^{lasso} = \operatorname{argmin}(y - X\beta)^T(y - X\beta)$$

$$\text{subject to } \underbrace{\sum \left|\beta_j\right|}_{\text{L1 norm}} \le s$$

By convention, the bias/intercept term is typically not regularized.
Here we assume data has been centered … therefore no bias term

Lasso Estimator



Ridge Regression

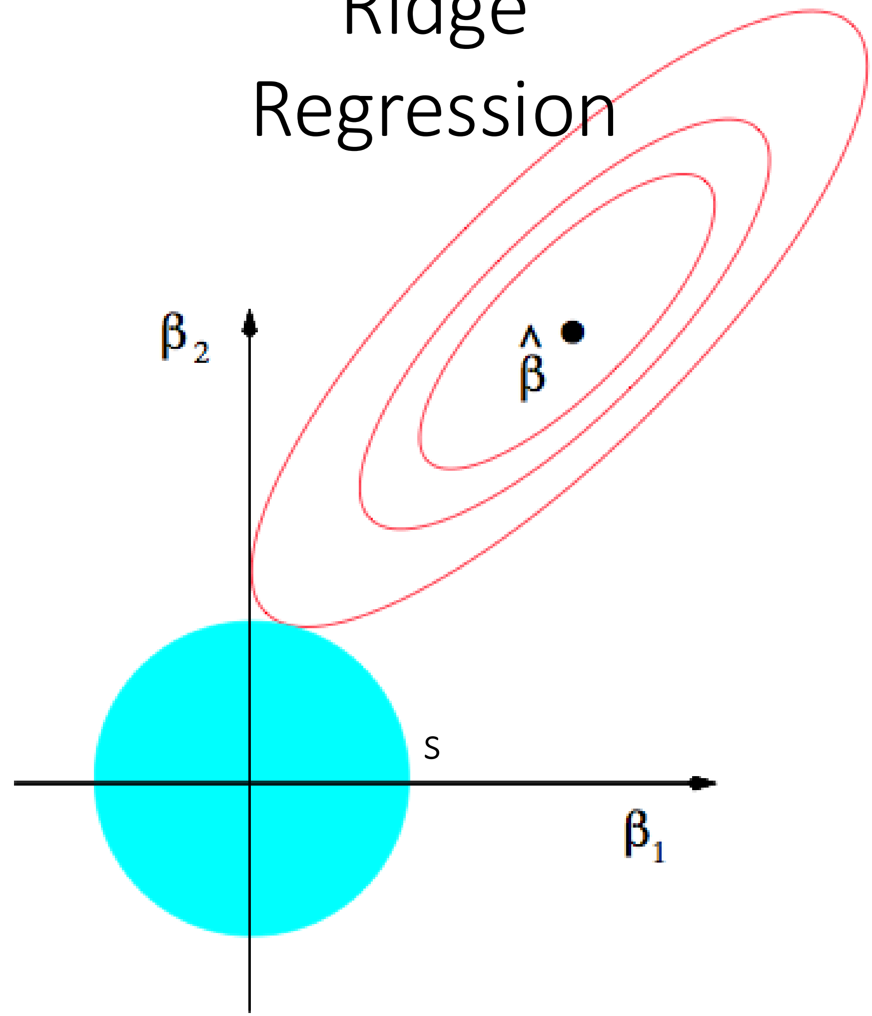


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

$$\min J(\beta) = \sum_{i=1}^{n}\left(Y - \hat{Y}\right)^2 + \lambda\left(\sum_{j=1}^{p}\beta_j^q\right)^{1/q}$$

Today: Regularized multivariate linear regression

Data: X

Task: y

Representation: : x, f()

Score Function: L()

Search/Optimization : argmin()

Models, Parameters

X: Tabular

Regression: y continuous

Y = Weighted linear sum of Xs

Sum of Squared Error (Least Squared) + Regularization

Revised Normal Equation / revised GD / SGD

Regression weights and bias => Regularized: Robust / Interpretable

More: A family of shrinkage estimators

$$\beta = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - x_i^T \beta)^2$$

$$\text{subject to } \sum |\beta_j|^q \le s$$

- for q >=0, contours of constant value of $\sum_j |\beta_j|^q$ are shown for the case of two inputs.

convex

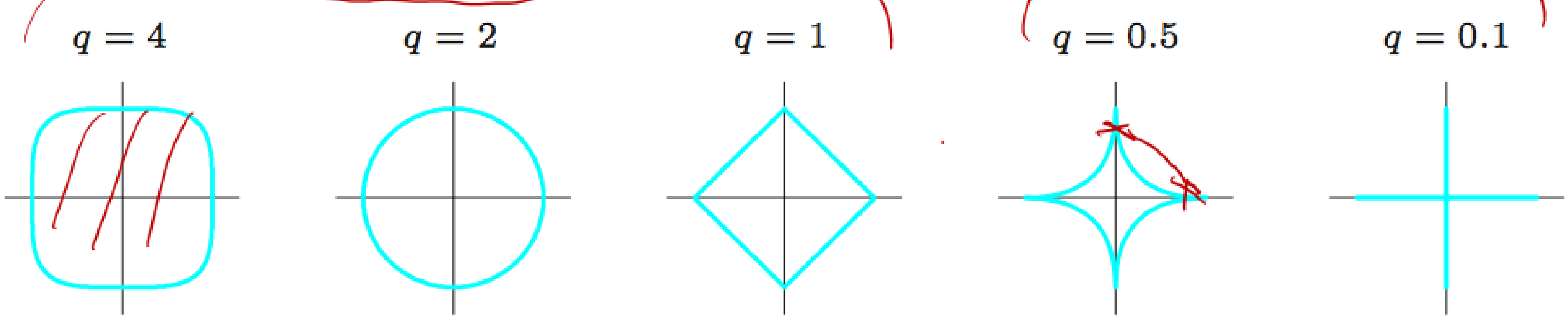
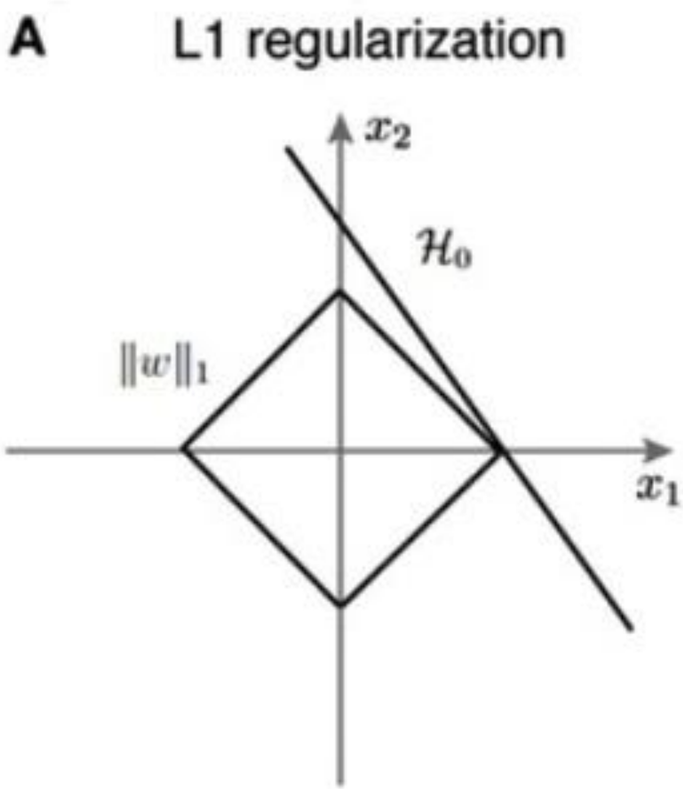


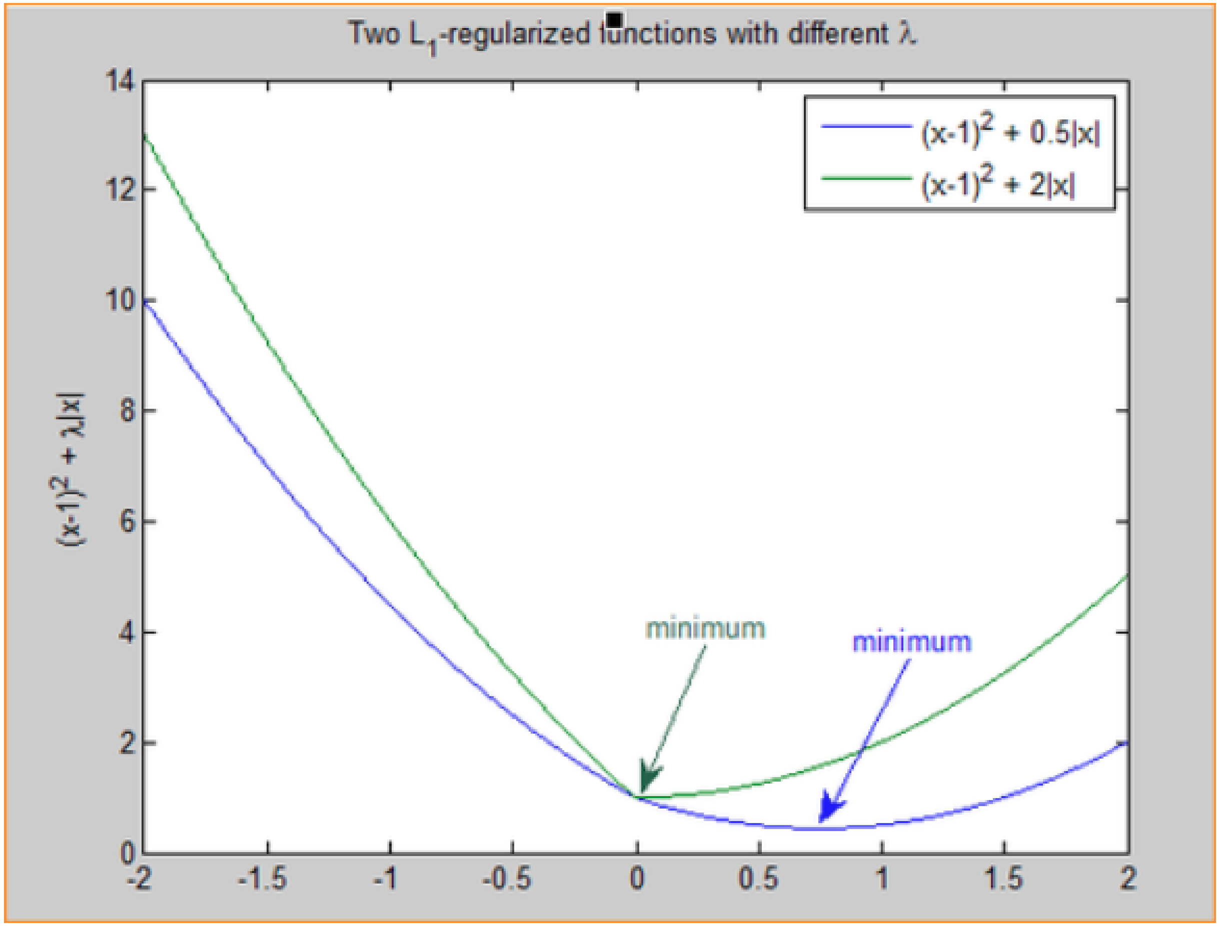
FIGURE 3.12. *Contours of constant value of $\sum_j |\beta_j|^q$ for given values of q.*



due to the nature of L_1 norm, the viable solutions are limited to corners, which are on a few axis only - in the above case x1. Value of x2 = 0. This means that the solution has eliminated the role of x2, leading to sparsity

L_1-regularized loss function $F(x) = f(x) + \lambda \|x\|_1$ is non-smooth. It's not differentiable at 0. Optimization theory says that the optimum of a function is either the point with 0-derivative or one of the irregularities (corners, kinks, etc.). So, it's possible that the optimal point of F is 0 even if 0 isn't the stationary point of f. In fact, it would be 0 if λ is large enough (stronger regularization effect). Below is a graphical illustration.

http://www.quora.com/What-is-the-difference-between-L1-and-L2-regularization



In mathematics, particularly in calculus, a stationary point or critical point of a differentiable function of one variable is a point of the domain of the function where the derivative is zero (equivalently, the slope of the graph at that point is zero).

Coordinate descent based Learning of Lasso

1. Initialize β
2. Repeat until converged
3. For $j = 1, 2, \ldots, P$ do

$$a_j = 2\sum_{i=1}^{n} x_{ij}^2$$

$$c_j = 2\sum_{i=1}^{n} x_{ij}\left(y_i - x_i^T\beta + x_{ij}\beta_j\right)$$

if $c_j < -\lambda$

$$\beta_j = (c_j + \lambda)/a_j$$

else if, $c_j > \lambda$

$$\beta_j = (c_j - \lambda)/a_j$$

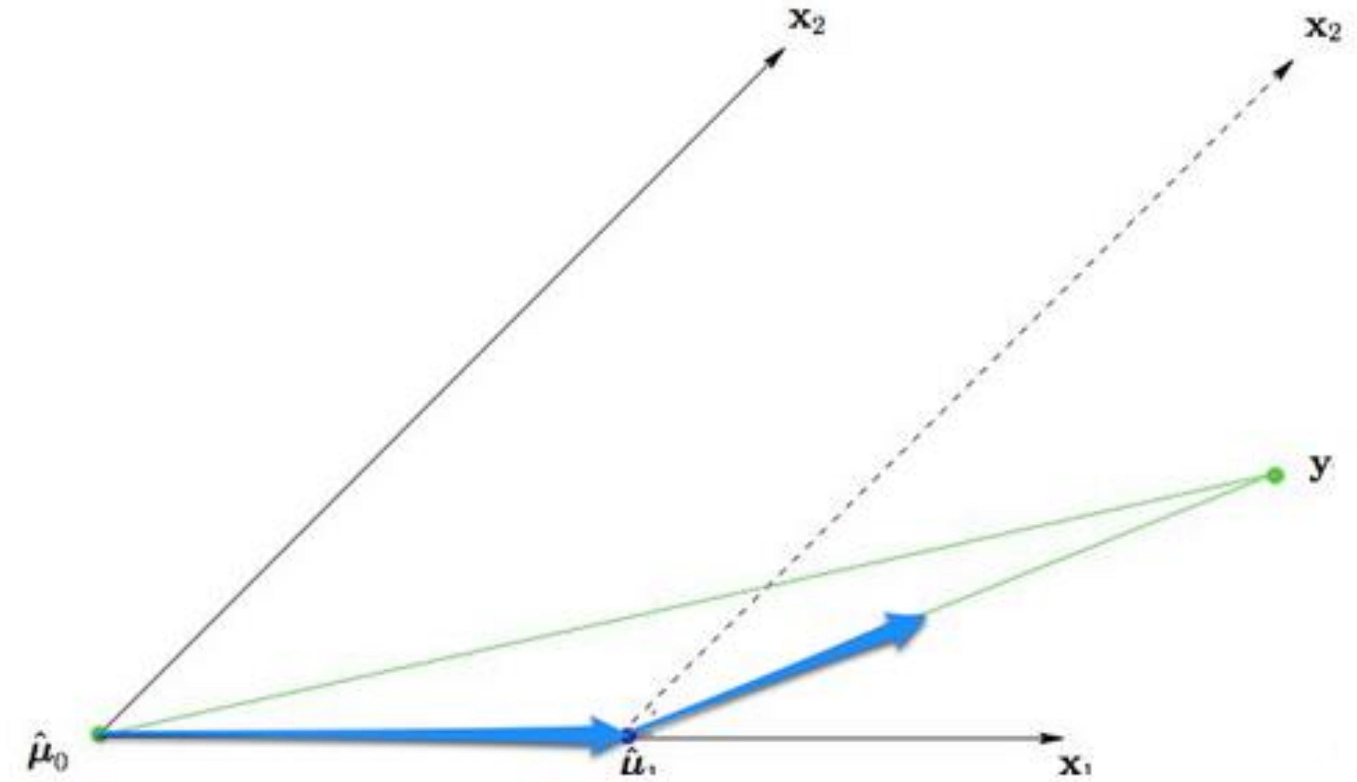
else, soft-thresholding

$$\beta_j = 0$$

Coordinate descent (WIKI)➔ one does line search along one coordinate direction at the current point in each iteration.

One uses different coordinate directions cyclically throughout the procedure.

Least Angle Regression (LARS) (State-of-the-art LASSO solver)



http://statweb.stanford.edu/~tibs/ftp/lars.pdf

L8: K-nearest-neighbor(regressor or classifier)

Data: X

↓

Task: y

↓

Representation: : x, f()

↓

Score Function: L()

↓

Search/Optimization : argmin()

↓

Models, Parameters

X: Tabular

↓

Classification or regression

↓

Local Smoothness / Training Samples

↓

EPE with L2 loss → conditional mean (Extra)

↓

NA

↓

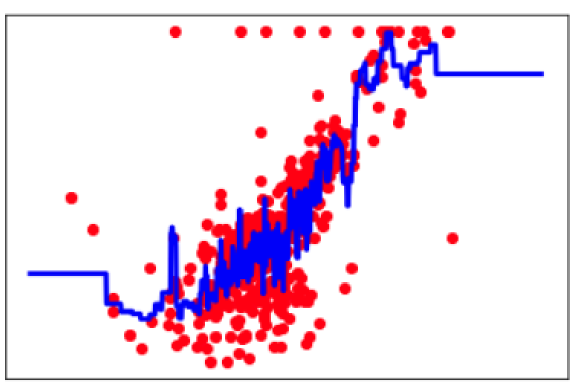
Training Samples

Code run: https://github.com/qiyanjun/2025Fall-UVA-CS-MachineLearningDeep/blob/main/notebook/L8_Knearest.ipynb

```
[42] # import regressor
     from sklearn.neighbors import KNeighborsRegressor
     # instantiate with K=5
     knn = KNeighborsRegressor(n_neighbors=5)
     # fit with data
     knn.fit(X, y)
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=3
                    metric_params=None, n_jobs=No
                    weights='uniform')
```

```
###
Xfit = np.linspace(3, 10, 1000).reshape(-1, 1)
yfit =knn.predict(Xfit)

# Plot outputs
plt.scatter(X, y,  color='red')
plt.plot(Xfit, yfit, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```

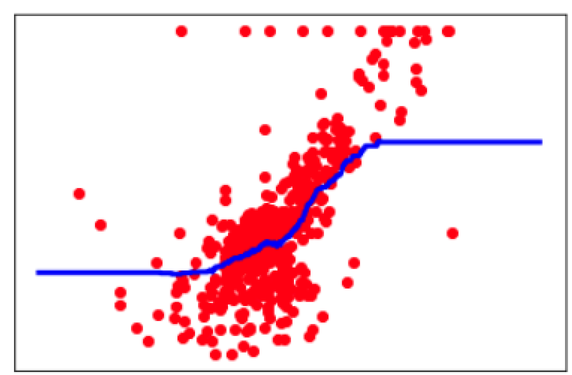


A CS

```
[44] # import regressor
     from sklearn.neighbors import KNeighborsRegressor
     # instantiate with K=5
     knn = KNeighborsRegressor(n_neighbors=100)
     # fit with data
     knn.fit(X, y)
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=3(
                    metric_params=None, n_jobs=Nor
                    weights='uniform')
```

```
###
Xfit = np.linspace(3, 10, 1000).reshape(-1, 1)
yfit =knn.predict(Xfit)

# Plot outputs
plt.scatter(X, y,  color='red')
plt.plot(Xfit, yfit, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```



K Nearest neighbor (Testing Mode)

It Needs:

1. The set of stored training samples
2. Distance metric to compute distance between samples
3. The value of k, i.e., the number of nearest neighbors to retrieve

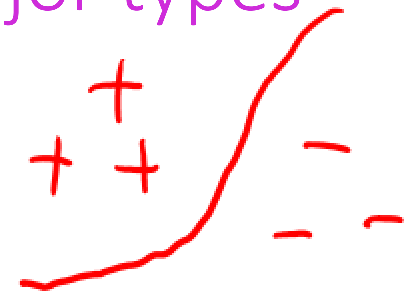
Training Mode:

- (Naïve) version: DO NOTHING !!!!

Testing Model: To classify unknown sample:

- Step1: Compute distance to all training records
- Step2: Identify k nearest neighbors
- Step3: Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

We can divide the large variety of supervised classifiers into roughly three major types

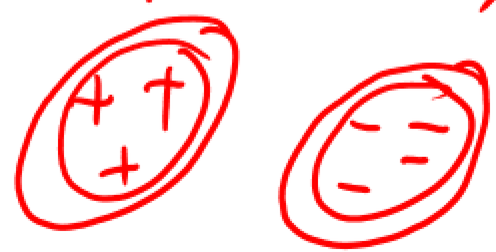


1. Discriminative
   directly estimate a decision rule/boundary
   e.g., support vector machine, decision tree,
   e.g. logistic regression, neural networks (NN), deep NN

$p(y|x)$

2. Generative:
   build a generative statistical model
   e.g., Bayesian networks, Naïve Bayes classifier

$p(x|y=c)$



3. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K nearest neighbors

Less complex $\longleftrightarrow$ More complex

		Less complex	More complex
① polynomial Regression	d	$d \downarrow$	$d \uparrow$
② Ridge Reg	$\underbrace{\lambda}_{>0}$	$\lambda \uparrow$	$\lambda \downarrow$
③ KNN Reg	K	$K \uparrow$	$K \downarrow$

Model Selection for Nearest neighbor classification

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

- Bias and variance tradeoff
  - A small neighborhood → large variance → unreliable estimation   overfit
  - A large neighborhood → large bias → inaccurate estimation   underfit

We aim to make our trained model

- 1. Generalize Well
- 2. Computational Scalable and Efficient
- 3. Trustworthy: Robust / Interpretable
  - Especially for some domains, this is about trust!

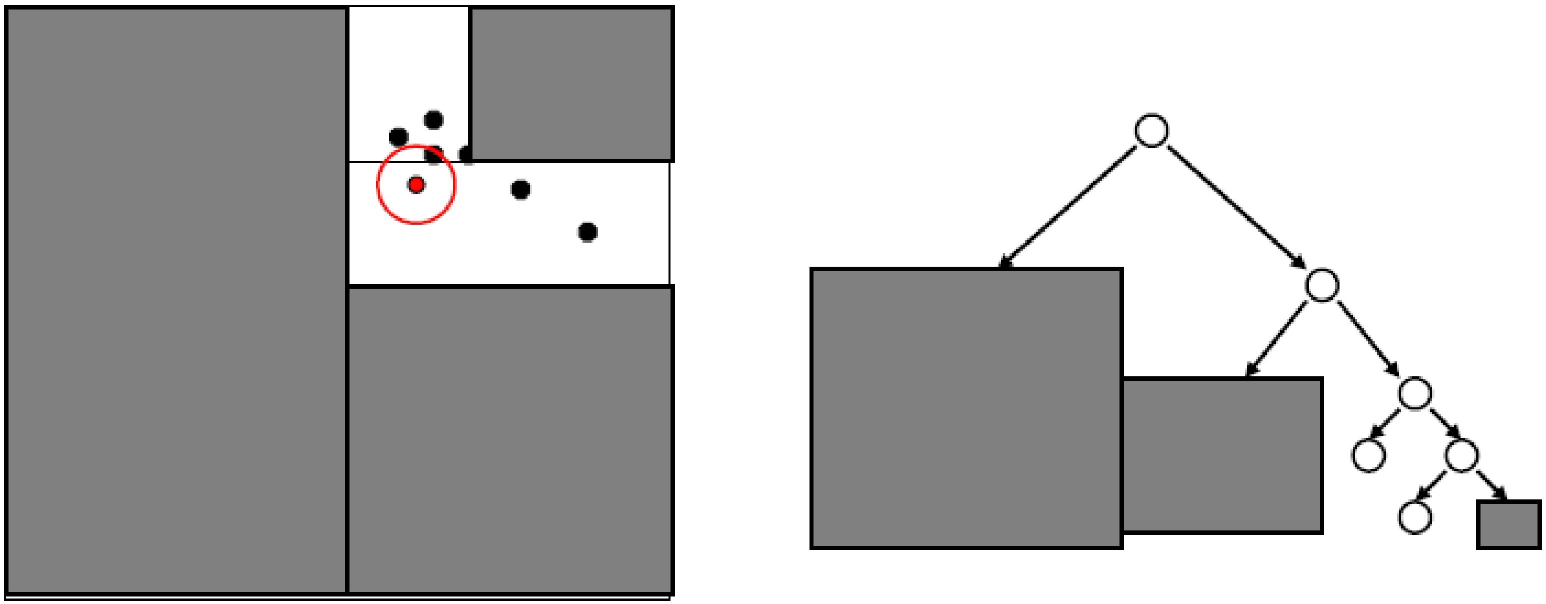
Computational Time Cost

	Train (n)	Test (m=1)
Linear Regression	$O(np^2+p^3)$	$O(p)$ $\hat{y}=\beta^T x$
KNN Reg	$O(1)$	$O(np)+$ $O(\text{sort } n\text{-}k)$???

$p = 30{,}000$

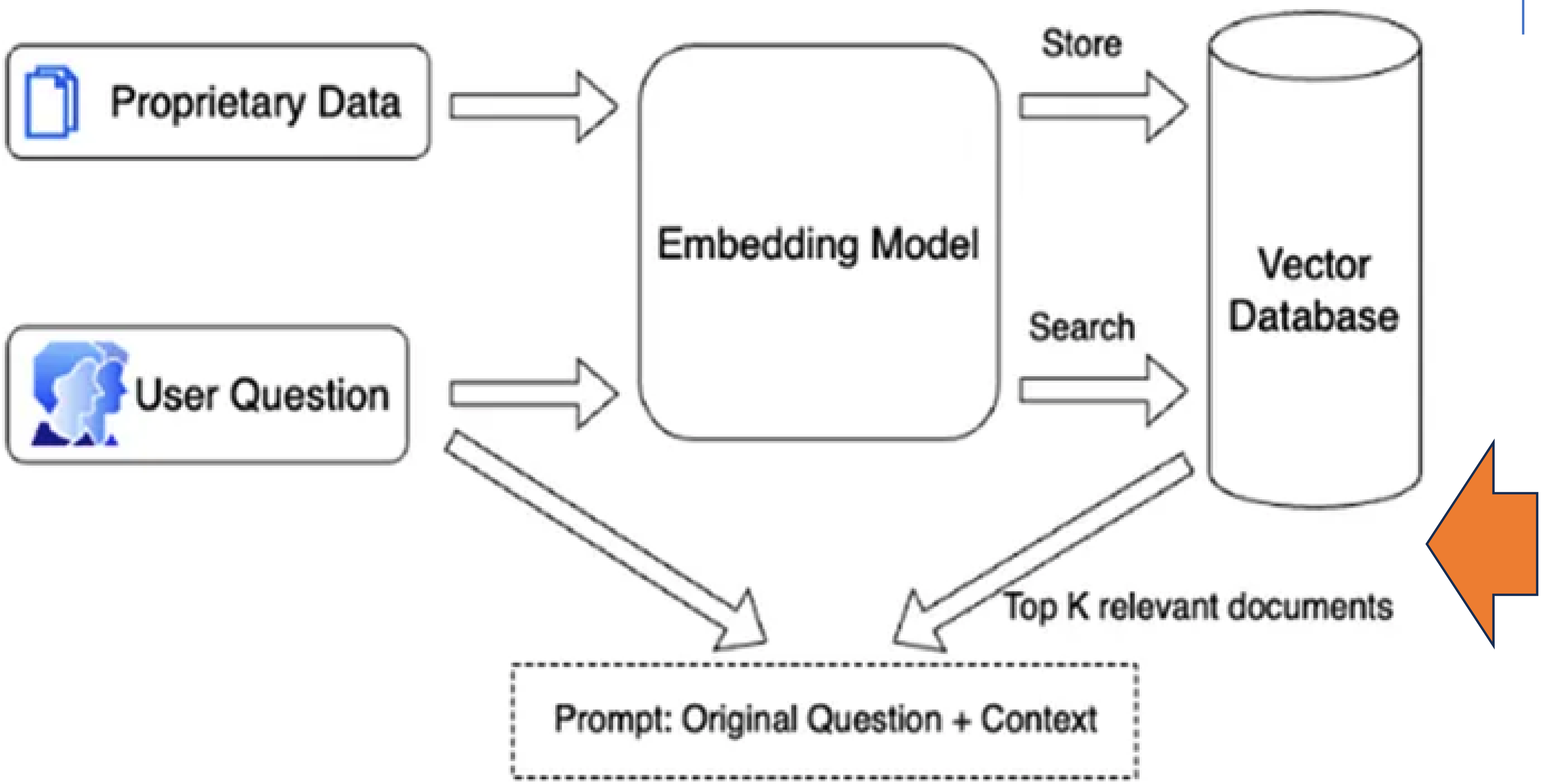
$n = 20{,}000$

NN Search by KD Tree



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

KNN as the Most critical component in Retrieval Augmented Generation System, e.g.:



09/30/2025 Roadmap

- TA to go over HW1
- One UVA ML club to introduce their setups and projects
- Q5
- Review Q4
- Review QA for L5-L7

10/07

10/07 /2025 Assignments

- HW2 is grading started ..
  - ➔ HW2 key walk-through next Thursday online zoom
- HW3 will get posted by tomorrow
  - Deep NN on Imaging task / Kera / mostly about learning modern DNN library
  - Programming + QA (like calculating marginal prob...)
- Next Tuesday is reading day
  - ➔ we will host makeup-Quiz Q7 next Thursday online
- Course format survey:
  - https://forms.gle/PkWGMkwHhawqf8QR8
  - Now go over the results:

Project Process

- Format:
  - Team (1~4 students)
  - Shark Tank alike Screening – https://en.wikipedia.org/wiki/Shark_Tank
  - This week: signup sheet for your team's screening sessions!
  - Next week: Initial project idea collecting!
  - TA Guangzhi will announce the process and signup sheet URL!

- Final deliverables:
  - (1) Code (Github PR to course project repo)
  - (2) Poster presentation class wide (Date: 12/09 TBD)
  - (3) Video Demo (after final exam)

10/07 /2025 Roadmp

- Review L5-L8 questions
- Quick Review L9-L10
- Review Q5
- Then Q6

quite disturbing for staying students around the right after quiz period

So we will host quiz after review / before project screening for all coming in-person sessions

Questions on L5-L8

Set 1: Bias–Variance, Overfitting, and Model Complexity

•How do bias and variance contribute to generalization error, and how do we find the "sweet spot" without knowing the true distribution?

•What are practical indicators of underfitting vs. overfitting (from graphs, learning curves, or error plots), and how do we fix each?

•How does cross-validation (choice of K) approximate generalization error, and what are the trade-offs (bias vs variance, LOOCV vs k-fold)?

•Why does zero training error often generalize poorly, and how is this linked to variance?

•Would we ever prefer high bias or high variance, and how do we reduce one while controlling the other?

Set 2: Regularization (LASSO, Ridge, Elastic Net, Generalizations)

•What are the key differences between L1 (LASSO), L2 (Ridge), and Elastic Net in terms of sparsity, robustness, computational cost, and when to use each?

•Why do L1 penalties set coefficients to zero, while L2 does not? What happens when p>np>n or when features are highly correlated?

•How does the choice of λ affect bias–variance, and how do we select it (cross-validation, validation curves)?

•Are there equivalent closed-form solutions for LASSO like Ridge has? Why are L1 and L2 chosen—what about higher-order penalties?

•When is Elastic Net preferable (e.g., grouped correlated features), and can we always default to it?

Questions on L5-L8

Set 3: k-Nearest Neighbors (kNN) and Instance-Based Learning

- How do we pick the best k (odd vs even, weighted vs unweighted, trade-offs with noisy data)?
- What is the computational cost of kNN (sorting term, memory cost), and can it overfit?
- How does the distance metric affect performance, and how are ties handled in classification?
- What are the advantages/disadvantages of kNN vs gradient descent or regularized linear models?
- In practice, how large must the dataset be to offset outliers, and is kNN more effective for regression or classification?

Set 4: Maximum Likelihood Estimation (MLE) and Probability Foundations

- Why do we usually maximize the log-likelihood instead of the likelihood itself, and how does this connect to squared error in linear regression?
- How does MLE extend from discrete distributions (e.g., coin flips) to continuous (e.g., Gaussians)?
- Why is the MLE for Bernoulli just the sample proportion, and what happens with small samples or noisy data?
- What makes MLE consistent and efficient, and are there situations where maximum likelihood may not yield the most “ideal” parameter?
- How does the bias–variance decomposition change with different loss functions (e.g., 0–1 loss, Laplace errors)?

10/07 /2025 Roadmp

- Review L5-L8 questions
- Quick Review L9-L10
- Review Q5
- Then Q6

quite disturbing for staying students around the right after quiz period

So we will host quiz after review / before project screening for all coming in-person sessions

Lecture 10: Maximum Likelihood Estimation (MLE)
→ Probability Review

- The big picture
- Events and Event spaces
- Random variables
- Joint probability, Marginalization, conditioning, chain rule, Bayes Rule, law of total probability, etc.
- Structural properties, e.g., Independence, conditional independence
- Maximum Likelihood Estimation

If hard to directly estimate from data, most likely we can estimate

- 1. Joint probability
  - Use Chain Rule

$$P(A, B) = P(B)\,P(A|B)$$

- 2. Marginal probability
  - Use the total law of probability

$$P(B) = P(B, A) + P(B, \sim A)$$
$$\| \qquad\qquad\qquad \|$$
$$P(B, A \cup \sim A)$$

- 3. Conditional probability
  - Use the Bayes Rule

$$P(A|B)$$
$$P(B|A) = \frac{P(A, B)}{P(A)} = \frac{P(A|B)\,P(B)}{P(A)}$$

One Example

Assume we have a dark box with 3 red balls and 1 blue ball. That is, we have the **set** *{r,r,r,b}. What is the probability of drawing 2 red balls in the first 2 tries?*

$$\boldsymbol{P(B_1 = r, B_2 = r)} = \underbrace{P(B_1=r)}_{\frac{3}{4}} \underbrace{P(B_2=r|B_1=r)}_{\frac{2}{3}} = \frac{1}{2}$$

$$P(B_2 = r) = P(B_1=r, B_2=r) + P(B_1=b, B_2=r)$$

$$P(B_1 = r|B_2 = r) = \frac{P(B_1=r, B_2=r)}{P(B_2=r)}$$

MLE idea is to

- ✓ assume a particular model with unknown parameters, θ
- ✓ we can then define the probability of observing a given event conditional on a particular set of parameters. $P(Z_i|\theta)$
- ✓ We have observed a set of outcomes in the real world.
- ✓ It is then possible to choose a set of parameters which are most likely to have produced the observed results.

$$\hat{\theta} = \underset{\theta}{argmax} \; P\left(Z_1 \ldots Z_n|\theta\right)$$

Likelihood

This is maximum likelihood.

In most cases this scorer is both consistent and efficient.

$$log(L(\theta)) = \sum_{i=1}^{n} log(P(Z_i|\theta))$$

Log-Likelihood

It is often convenient to work with the Log of the likelihood function.

Deriving the Maximum Likelihood Estimate for Bernoulli

$$\log(L(p)) = \log\left[\prod_{i=1}^{n} p^{z_i}(1-p)^{1-z_i}\right]$$

$$= \sum_{i=1}^{n}(z_i \log p + (1-z_i)\log(1-p))$$

$$= \log p \sum_{i=1}^{n} z_i + \log(1-p)\sum_{i=1}^{n}(1-z_i)$$

$$= \mathrm{x}\log p + (\mathrm{n}-\mathrm{x})\log(1-p)$$

Observed data ➔ x
heads-up from n trials

Deriving the Maximum Likelihood Estimate for Bernoulli

$$\arg\min_p \{-l(p)\} = \arg\min_p \{-x\log(p)-(n-x)\log(1-p)\}$$

$$\frac{dl(p)}{dp} = -\frac{x}{p}-\frac{-(n-x)}{1-p} = 0$$

$$0=-\frac{x}{p}+\frac{n-x}{1-p}$$

$$0=\frac{-x(1-p)+p(n-x)}{p(1-p)}$$

$$0=-x+px+pn-px$$

$$0=-x+pn$$

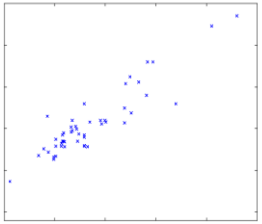
Minimize the negative log-likelihood

➔ MLE parameter estimation

$$\hat{p}=\frac{x}{n}$$

i.e. Relative frequency of a binary event

DETOUR: Probabilistic Interpretation of Linear Regression



- Let us assume that the target variable and the inputs are related by the equation:

RV $\varepsilon \sim N(0, \sigma^2)$

$$y_i = \theta^T \mathbf{x}_i + \varepsilon_i$$

where ε is an error term of unmodeled effects or random noise

2

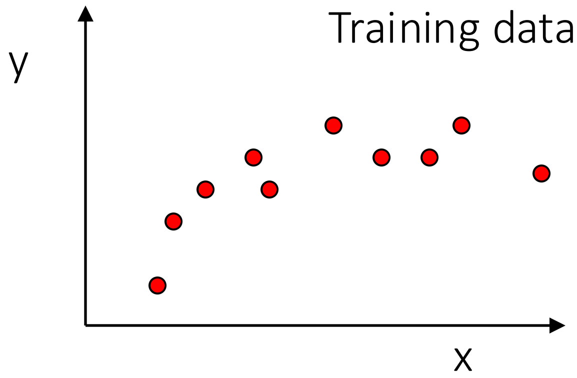
- Now assume that ε follows a Gaussian N(0,σ), then we have:

$$p(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2} \right)$$

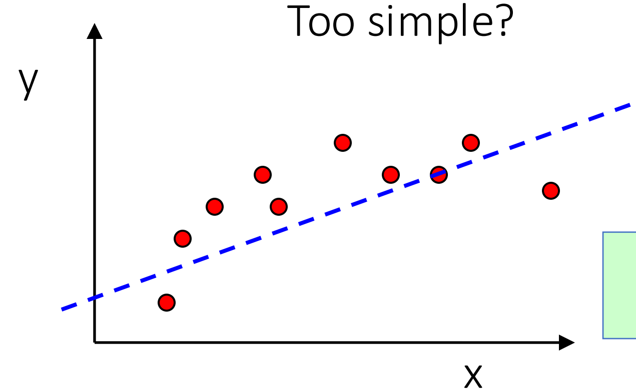
RV $y|x;\theta \sim N(\theta^T x, \sigma)$

L9: Complexity / Goodness of Fit / Generalization

Training data



Too simple?

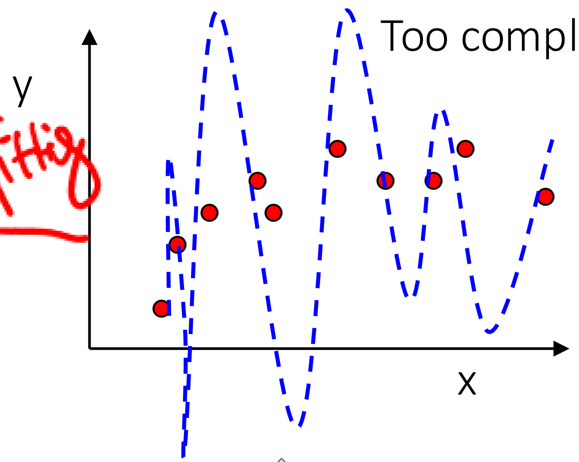


under-fit

High Bias

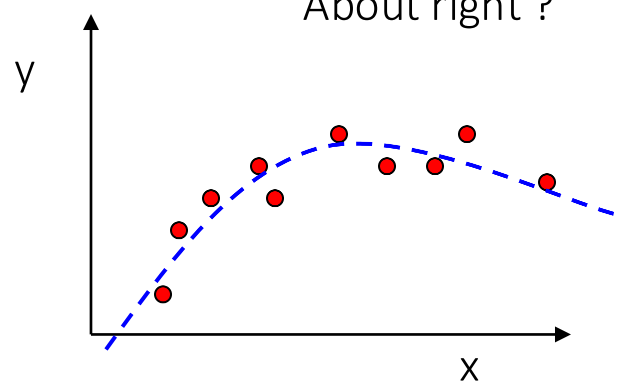
Too complex ?

overfitting



High Variance

About right ?



What ultimately matters: GENERALIZATION

Statistical Decision Theory (Extra)

- Random input vector: X
- Random output variable: Y
- Joint distribution: Pr(X,Y)
- Loss function L(Y, f(X))

$P(t_1, t_2): \begin{cases} HH \\ HT \\ TH \\ TT \end{cases}$

$\Rightarrow D = \boxed{\begin{matrix} (\vec{x}_1, y_1) \\ \vdots \\ (\vec{x}_n, y_n) \end{matrix}}$

- Expected prediction error (EPE):

$$\mathrm{EPE}(f) = \mathrm{E}(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$

$$\text{e.g.} = \int (y - f(x))^2 \boxed{\Pr(dx, dy)}$$

e.g. Squared error loss (also called L2 loss)

One way to define generalization: by considering the joint population distribution

Extra

Decomposition of EPE

- When additive error model: $Y = f(X) + \epsilon,\ \epsilon \sim (0, \sigma^2)$
- Notations
  - Output random variable: Y
  - True function: f → true
  - Prediction estimator: $\hat{f}$ → D → $\hat{f}$

$$
\begin{aligned}
EPE(x) &= E[(Y - \hat{f})^2 | X = x] \\
&= E[((Y - f) + (f - \hat{f}))^2 | X = x] \\
&= \underbrace{E[\underbrace{(Y - f)}_{\epsilon}{}^2 | X = x]}_{\text{Bayes Error}} + \underbrace{E[(f - \hat{f})^2 | X = x]} \\
&= \sigma^2 + \underbrace{Var(\hat{f}) + \text{Bias}^2(\hat{f})}
\end{aligned}
$$

Irreducible / Bayes error

Bias-Variance Trade-off for EPE:

$$EPE(x) = noise^2 + bias^2 + variance$$

Unavoidable error

Error due to incorrect assumptions

Error due to variance of training samples

Slide credit: D. Hoiem

Extra, But VERY IMPORTANT

BIAS AND VARIANCE TRADE-OFF for Parameter Estimation

- θ: true value (normally unknown)
- $\hat{\theta}$: estimator
- $\bar{\theta} := E[\hat{\theta}]$ (mean, i.e. expectation of the estimator)

- Bias $E[(\bar{\theta} - \theta)^2]$
  - measures accuracy or quality of the estimator
  - low bias implies on average we will accurately estimate true parameter from training data

- Variance $E[(\hat{\theta} - \bar{\theta})^2]$
  - Measures precision or specificity of the estimator
  - Low variance implies the estimator does not change much as the training set varies

Model "bias" & Model "variance"

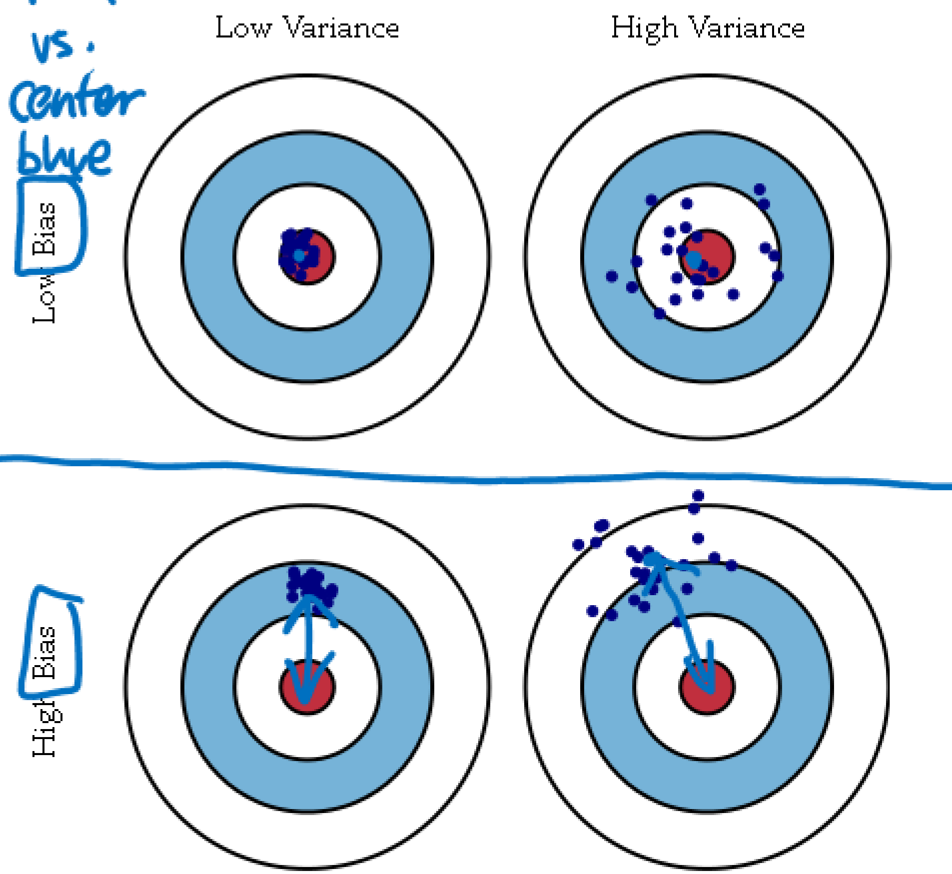
- Middle RED:
  - TRUE function
- Error due to bias:
  - How far off in general from the middle red

$$E[(\bar{\theta} - \theta)^2]$$

- Error due to variance:
  - How wildly the blue points spread

$$E[(\hat{\theta} - \bar{\theta})^2]$$

red vs. center blue



need to make assumptions that are able to generalize

- Underfitting: model is too "simple" to represent all the relevant characteristics
  - High bias and low variance
  - High training error and high test error

- Overfitting: model is too "complex" and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

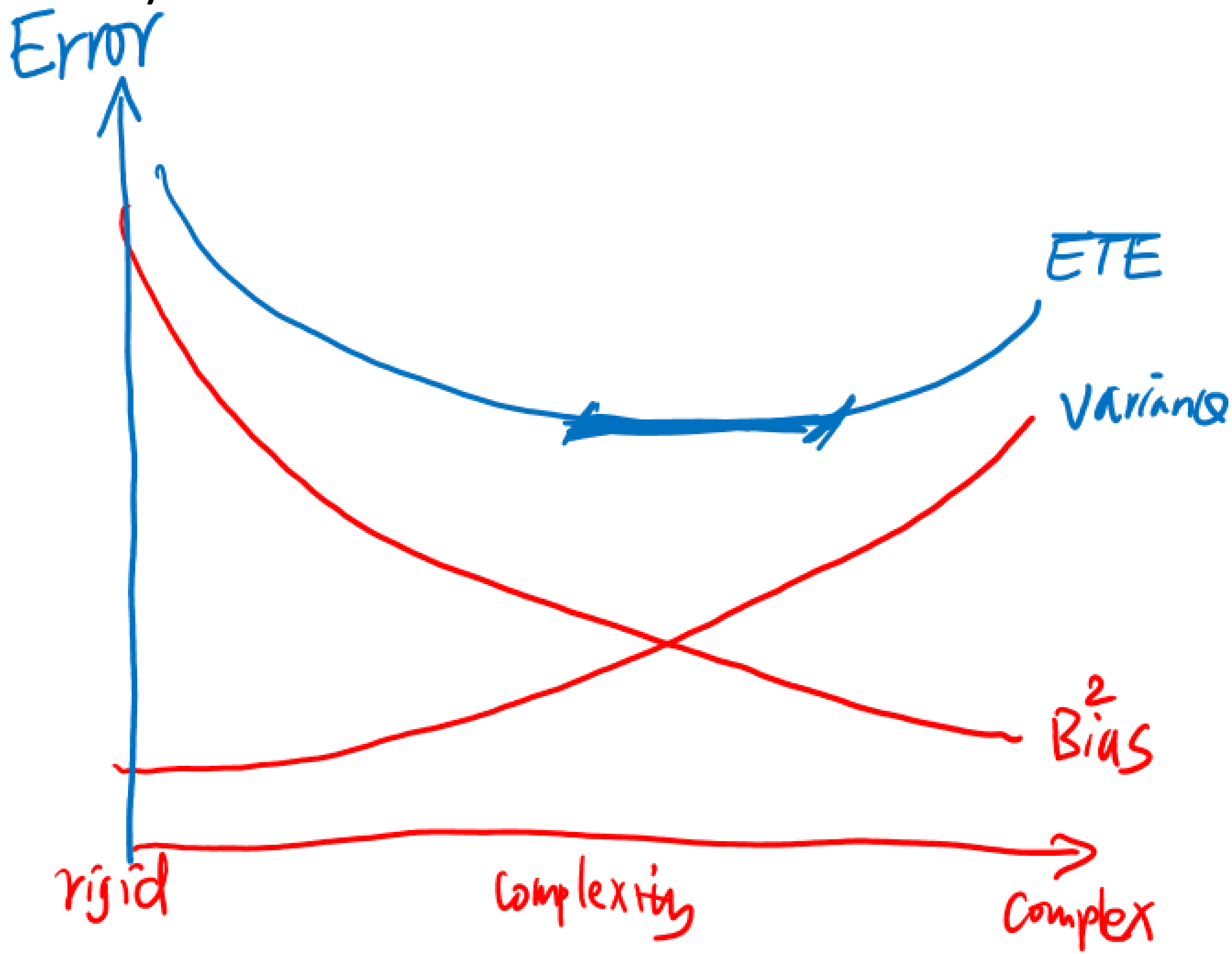
Slide credit: L. Lazebnik

Bias Variance Tradeoff

- (1) Randomness of Training Sets
- (2) Training error can always be reduced when increasing model complexity
- (3) Randomness in the Testing Error!!!
- (4) Cross Validation Error as good approximation for Expected Test error -- good appx of generalization

Review:
One important Control of Bias Variance Tradeoff
➔ Model Complexity

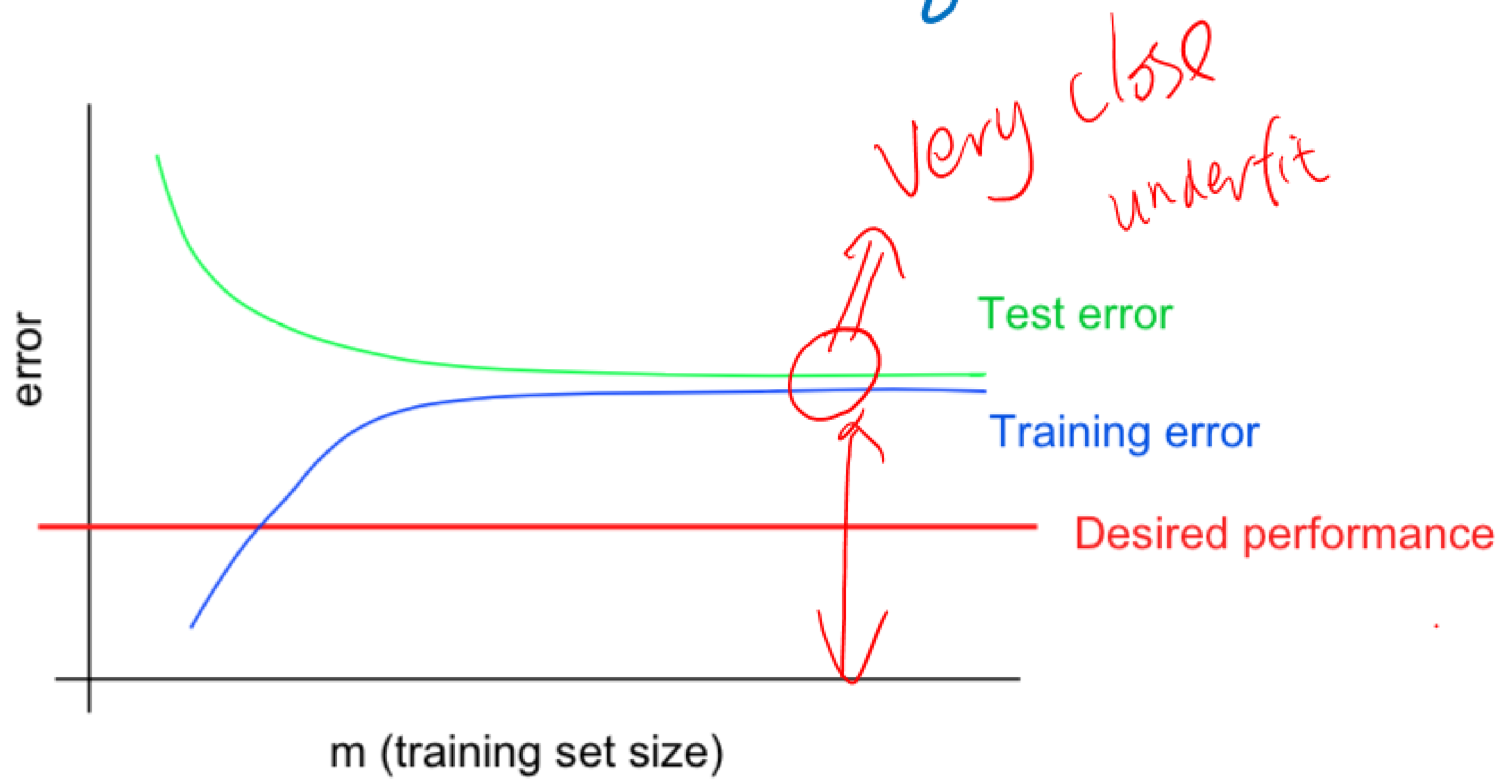
- bias decrease with model gets more complex;
- Variance increase with bigger model capacity
- Sum of Bias^2+Variance



Another important Control of Bias Variance Tradeoff ➔ Training Size (Extra)

Typical learning curve for high bias:

learning curve



- Even training error is unacceptably high.
- Small gap between training and test error.

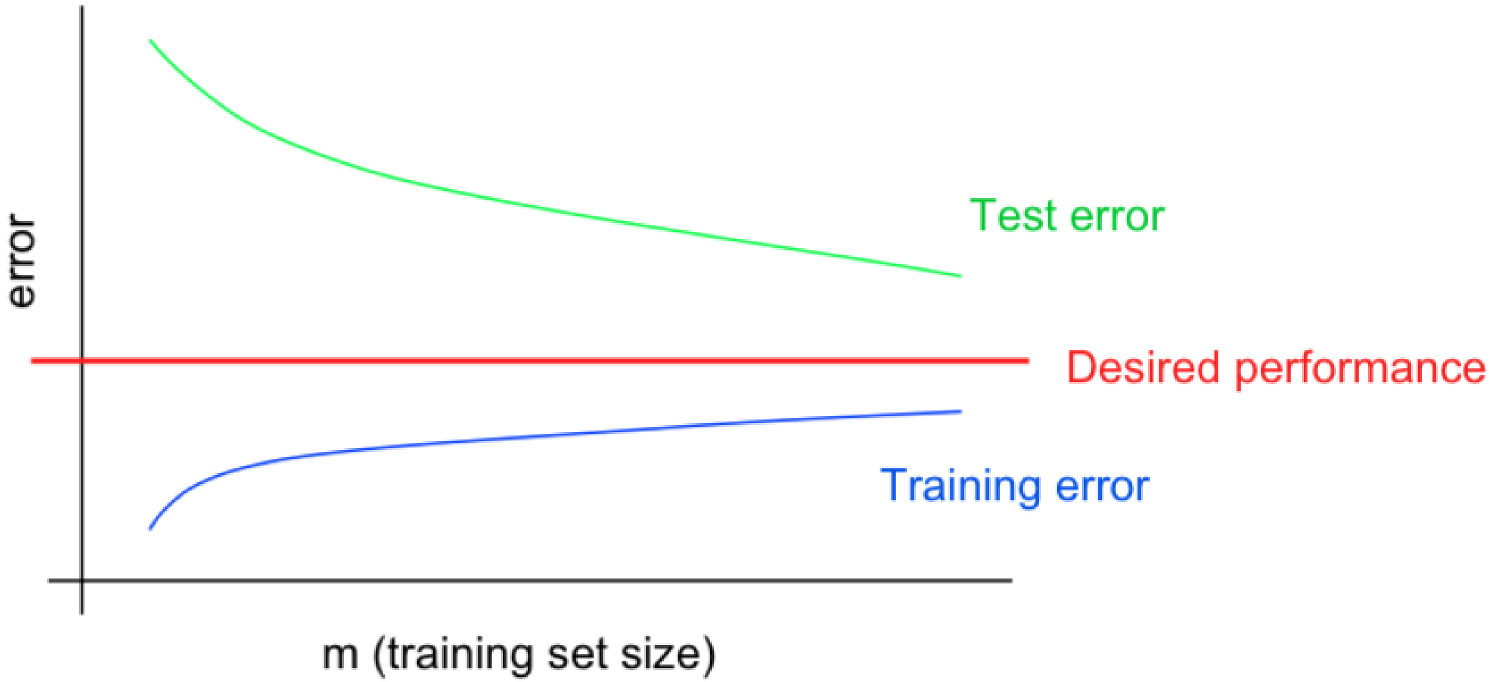
High training error and high test error

Slide credit: A. Ng

Another important Control of Bias Variance Tradeoff ➔ Training Size (Extra)

Typical learning curve for high variance:

learning curve



Slide credit: A. Ng

How to reduce Model High Variance?

- Choose a simpler classifier
- Regularize the parameters
- Get more training data
- Try smaller set of features feature Selection $n < p$
- Try feature engineering
- Try multiple models and then use all as ensemble

Slide credit: D. Hoiem

Take Away : Three types of plots

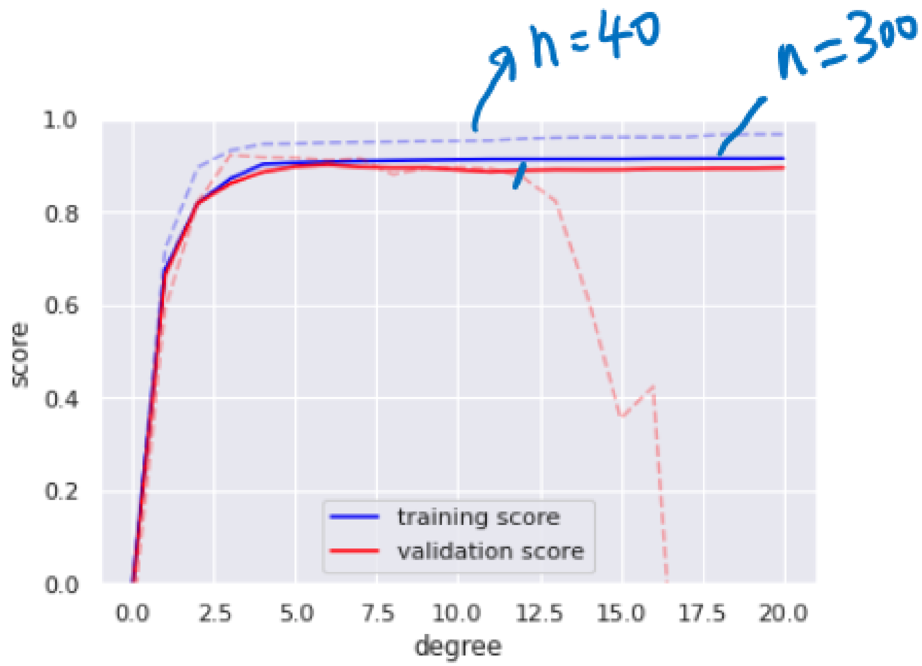
- (1) Sanity check (S)GD type Optimization
  - Train / Vali Loss vs. Epochs to help you
    - https://scikit-learn.org/stable/auto_examples/linear_model/plot_sgd_early_stopping.html#sphx-glr-auto-examples-linear-model-plot-sgd-early-stopping-py

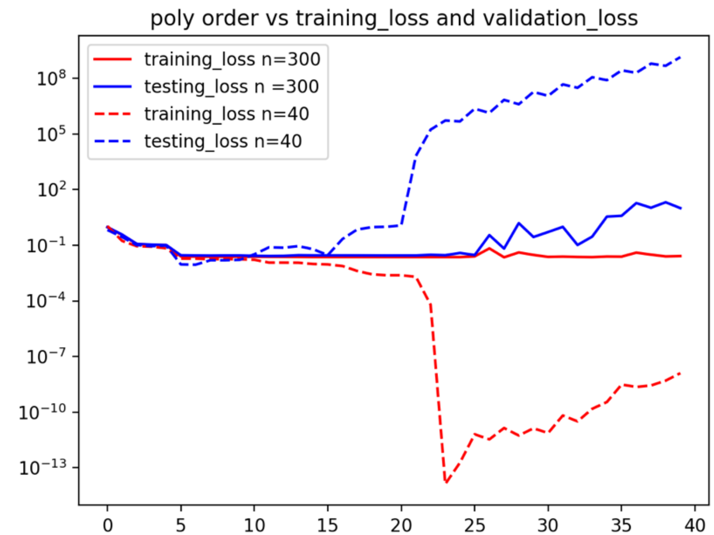
- (2) Sanity check hyperparameter tuning (validation curve)
  - Train / Vali Loss vs. hyperparameter Values

    from sklearn.model_selection import validation_curve

- (3) Sanity check if your current model overfits or underfits
  - Train / Vali Loss vs. Varying Size of Training (learning curve)
    - https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html#sphx-glr-auto-examples-model-selection-plot-learning-curve-py

I will Code run: https://github.com/qiyanjun/2025Fall-UVA-CS-MachineLearningDeep/blob/main/notebook/L9-LearningCurves.ipynb





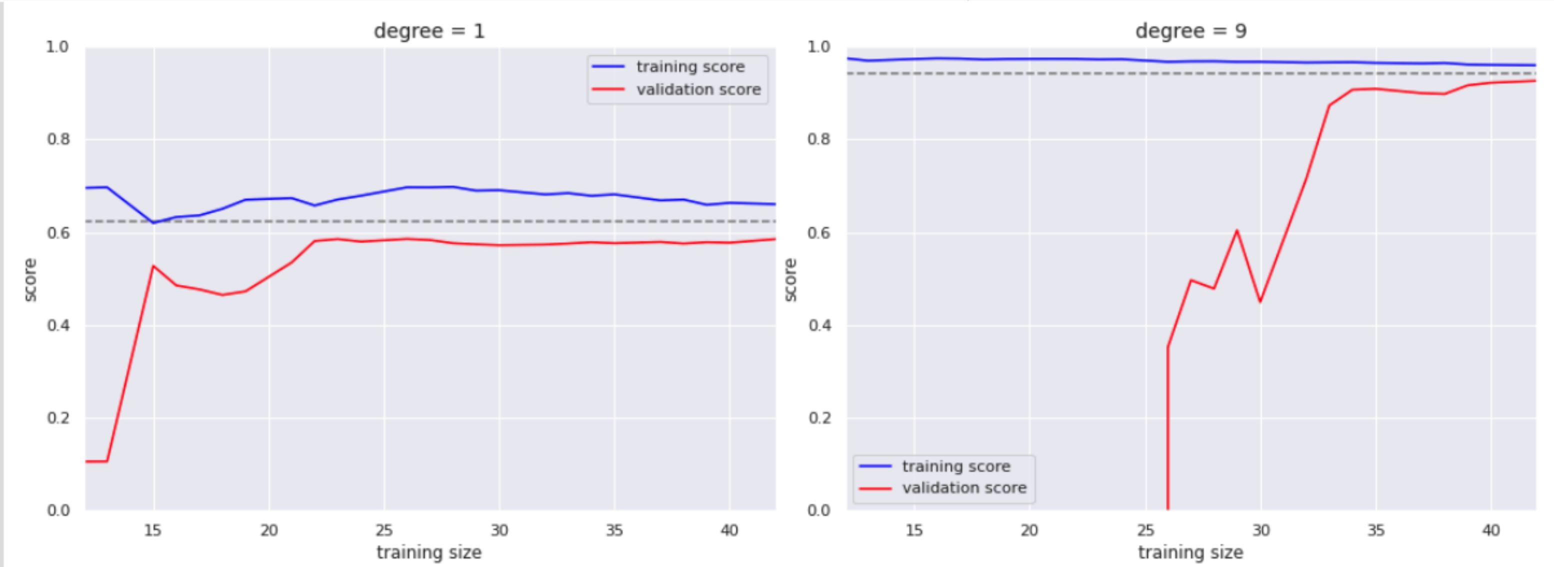
(1) Validation_curve

(1) Validation_curve

By scikitlearn Validation_curve function (normalize all metrics to positive range

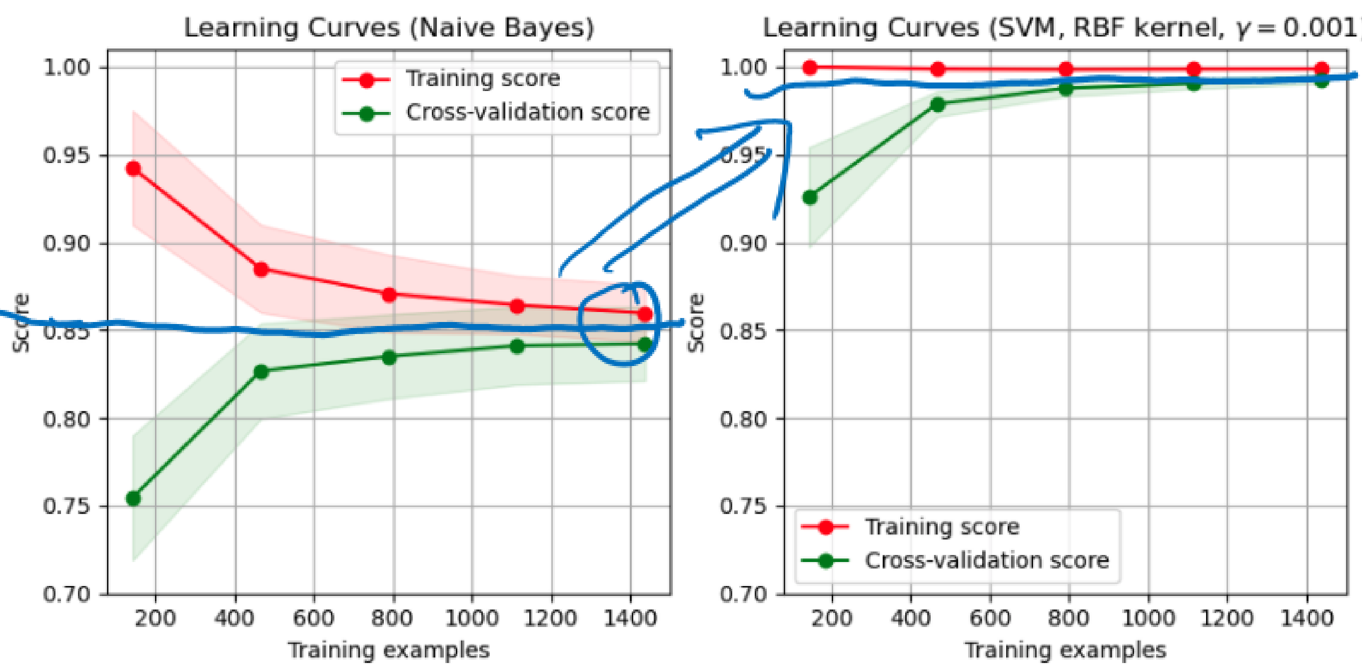
By our HW2 (more close to modern deep learning library style)

https://scikit-learn.org/stable/modules/model_evaluation.html#the-scoring-parameter-defining-model-evaluation-rules

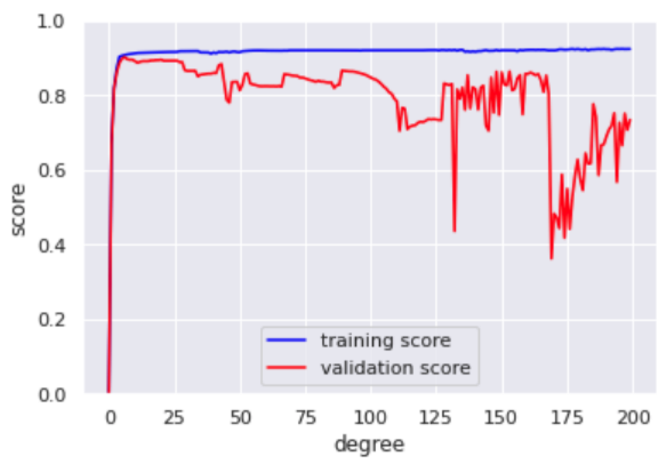


(1) Learning Curves for polynomial regression (up) and classification (down) / by scikitlearn

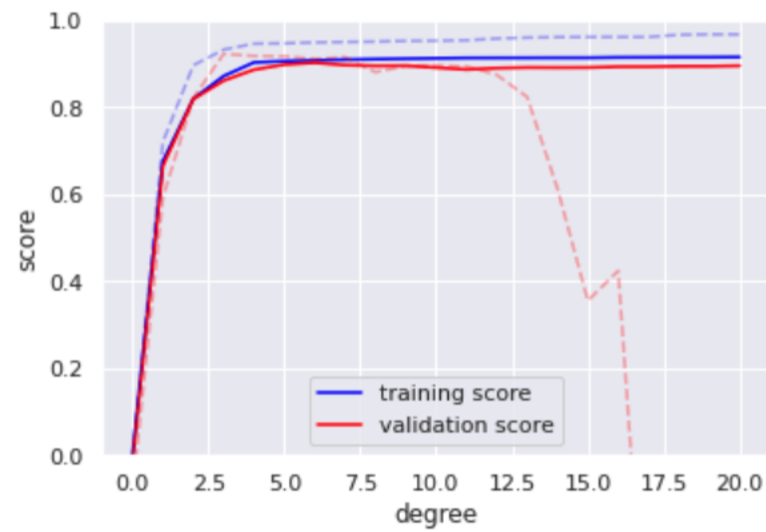
underfit



```
X2, y2 = make_data(200)

degree = np.arange(200)
train_score2, val_score2 = validation_curve(PolynomialR
                                             'polynomial

plt.plot(degree, np.median(train_score2, 1), color='blu
plt.plot(degree, np.median(val_score2, 1), color='red',
plt.legend(loc='lower center')
plt.ylim(0, 1)
plt.xlabel('degree')
plt.ylabel('score');
```




```
X2, y2 = make_data(200)
degree = np.arange(21)
train_score2, val_score2 = validation_curve(PolynomialReg
                                             'polynomialfe

plt.plot(degree, np.median(train_score2, 1), color='blue'
plt.plot(degree, np.median(val_score2, 1), color='red', l
plt.plot(degree, np.median(train_score, 1), color='blue',
plt.plot(degree, np.median(val_score, 1), color='red', al
plt.legend(loc='lower center')
plt.ylim(0, 1)
plt.xlabel('degree')
plt.ylabel('score');
```



Interesting Relation between

- the right range of model complexity
- the number of training points

Is the bias-variance trade off dependent on the number of samples? (EXTRA)

$n \nearrow$

variance $\swarrow$

In the usual application of linear regression, your coefficient estimators are unbiased so sample size is irrelevant. But more generally, you can have bias that is a function of sample size as in the case of the variance estimator obtained from applying the population variance formula to a sample (sum of squares divided by n).....

... the bias and variance for an estimator are generally a decreasing function of training size n. Dealing with this is a core topic in nonparametric statistics. For nonparametric methods with tuning parameters a very standard practice is to theoretically derive rates of convergence (as sample size goes to infinity) of the bias and variance as a function of the tuning parameter, and then you find the optimal (in terms of MSE) rate of convergence of the tuning parameter by balancing the rates of the bias and variance. Then you get asymptotic results of your estimator with the tuning parameter converging at that particular rate. Ideally you also provide a data-based method of choosing the tuning parameter (since simply setting the tuning parameter to some fixed function of sample size could have poor finite sample performance), and then show that the tuning parameter chosen this way attains the optimal rate.

https://www.reddit.com/r/statistics/comments/6uajyr/is_the_biasvariance_trade_off_dependent_on_the/

10/16

Agenda

- Going over HW2 Solution
- A tutorial talk on Huggingface.co

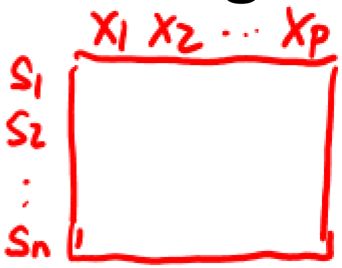
Course Content Plan ➔ Regarding Tasks

JID

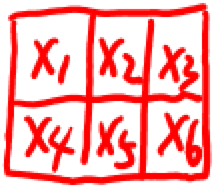
- ~~Regression (supervised)~~ — Y is a continuous
- ~~Learning theory~~ — About f()
- Classification (supervised) — Y is a discrete
- Unsupervised models — NO Y
- Graphical models — About interactions among Y,X1,. Xp
- Reinforcement Learning — Learn to Interact with environment

Course Content Plan ➔ Regarding Data

- ❑ Tabular / Matrix



- ❑ 2D Grid Structured: Imaging



- ❑ 1D Sequential Structured: Text
- ❑ Graph Structured (Relational)
- ❑ Set Structured / 3D /

Today: Logistic Regression **Classifier**

$$P(y=1|x)=\frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}}$$

Data: X

Task: y

Representation: : x, f()

Score Function: L()

Search/Optimization : argmin()

Models, Parameters, metrics

X: Tabular

Binary Classification

Log-odds y = linear function of Xs

MLE

Iterative (Newton) method / SGD

Logistic weights

/ Accuracy / F1

Bayes Classifiers – Predict via MAP Rule

Task: Classify a new instance X: $X = \langle X_1, X_2, \ldots, X_p \rangle$

based on:

$$c_{MAP} = \underbrace{\underset{c_j \in C}{\operatorname{argmax}}}\, P(c_j \mid x_1, x_2, \ldots, x_p)$$

MAP Rule

MAP = Maximum Aposteriori Probability

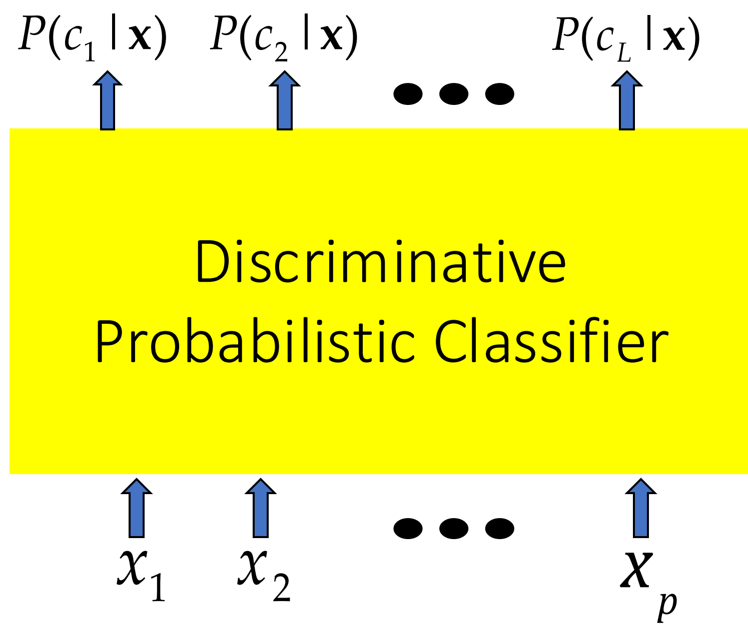
Please read extra slides for how to get MAP-rule from minimize EPE-0/1

Our Whole Section 2:

$X \longrightarrow C \quad : \quad \underbrace{P(C|X)}_{f(x)}$

– Discriminative Classifiers

$$\arg\max_{c \in C} P(c / \mathbf{X}),\quad C = \{c_1, \cdots, c_L\}$$



$P(c|\vec{x})$

$\uparrow$

$\vec{x}$

$$\mathbf{x} = (x_1, x_2, \cdots, x_p)$$

Adapt from Prof. Ke Chen NB slides

Today: Logistic Regression **Classifier**

$$P(y=1|x)=\frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}}$$

Data: X

Task: y

Representation: : x, f()

Score Function: L()

Search/Optimization : argmin()

Models, Parameters, metrics

X: Tabular

Binary Classification

Log-odds y = linear function of Xs

MLE

Iterative (Newton) method / SGD

Logistic weights / Accuracy / F1

$P(y=0|x) = 1 - P(y=1|x)$

Logistic Regression p(y|x)

$$\underbrace{\ln\left[\frac{P(y|x)}{1-P(y|x)}\right]}_{\text{Logit}} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

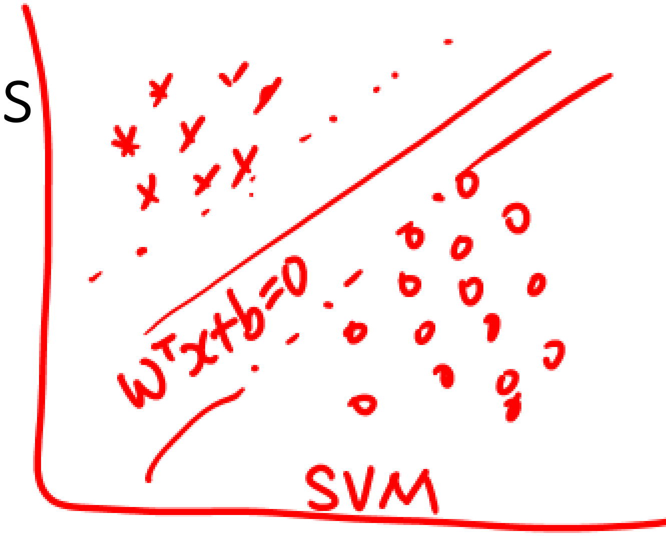
log [P(Head|x) / P(Tail|x)]

P(C|X)

P(Head)

$$= P(y|x) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\dots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\dots+\beta_p x_p}} = \frac{1}{1+e^{-(\beta_0+\beta^T X)}}$$

View IV: Logistic Regression models a linear classification boundary!



$$y = \{H, T\} = \{1, 0\}$$

$$\underset{y \in \{0,1\}}{\operatorname{argmax}}\; p(y \mid \underline{\underline{x}})$$

$$p(y=0 \mid x) = p(y=1 \mid x) \Rightarrow$$

Decision Boundary

$$\frac{p(y=1 \mid x)}{p(y=0 \mid x)} = 1$$

$$\log\left(\frac{p(y=1 \mid x)}{p(y=0 \mid x)}\right) = \vec{\beta}^T \vec{x} = \log(1) = 0$$

Extra

Summary: MLE for Logistic Regression Training

Let's fit the logistic regression model for K=2, i.e., number of classes is 2

Training set: (x_i, y_i), i=1,…,N

For Bernoulli distribution

$$p(y|x)^y(1-p)^{1-y}$$

(conditional)
Log-likelihood:

How?

$$l(\beta) = \sum_{i=1}^{N} \{\log \Pr(Y = y_i | X = x_i)\}$$

$$= \sum_{i=1}^{N} y_i \log(\Pr(Y=1|X=x_i)) + (1-y_i)\log(\Pr(Y=0|X=x_i))$$

$$= \sum_{i=1}^{N} (y_i \log \frac{\exp(\beta^T x_i)}{1+\exp(\beta^T x_i)}) + (1-y_i)\log \frac{1}{1+\exp(\beta^T x_i)})$$

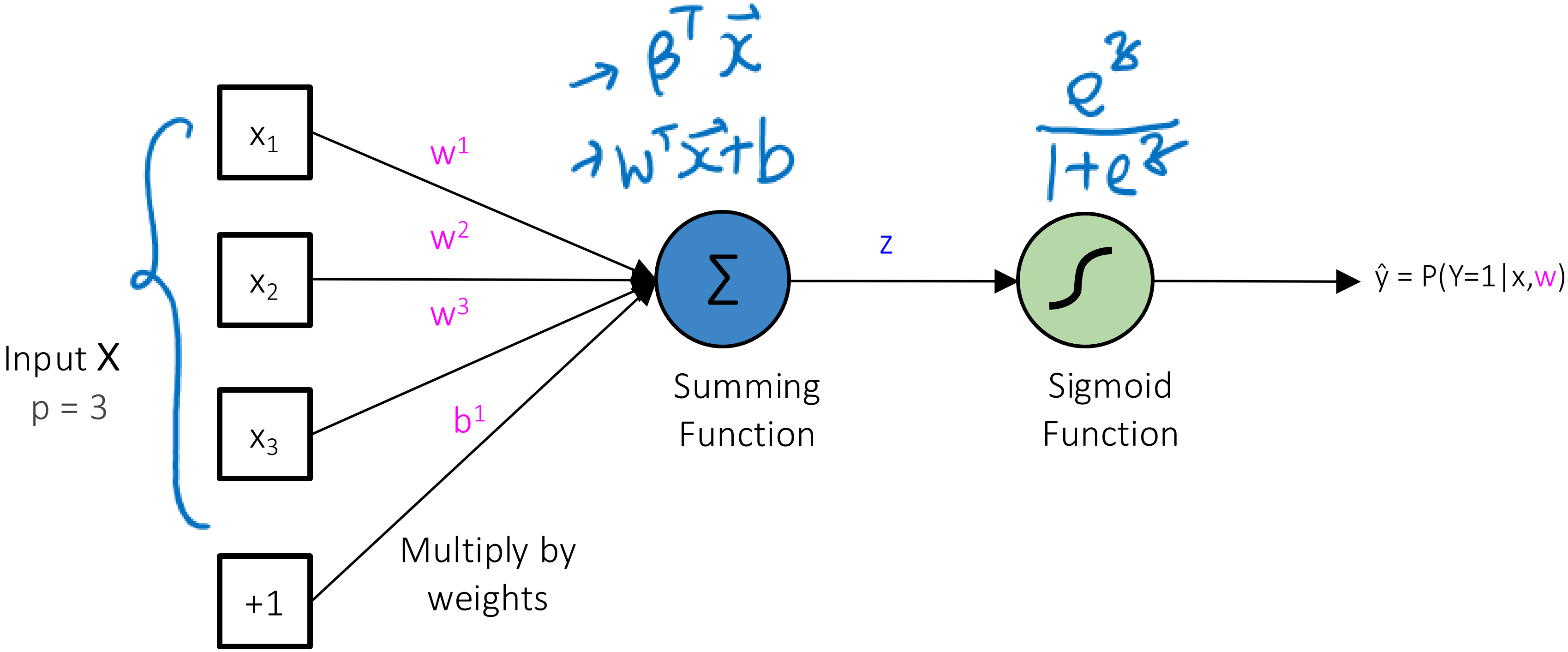
$$= \sum_{i=1}^{N} (y_i \beta^T x_i - \log(1+\exp(\beta^T x_i)))$$

x_i are (p+1)-dimensional input vector with leading entry 1
\beta is a (p+1)-dimensional vector

We want to maximize the log-likelihood in order to estimate \beta

See Extra Slides How to used Newton-Raphson optimization

One "Neuron": Block View of Logistic Regression



$z = W^T \cdot X + b$

$y = \text{sigmoid}(z) = \frac{e^z}{1 + e^z}$

10/21

Agenda

- HW3 is due
- Please select your Project's Shark Tank Sessions ASAP

- Today:
  - Review MLP / DNN / CNN / PCA / Word Embedding, Transformer
  - Quiz 8 Today

Takeaway: Logistic Regression **Classifier**

- View I: logit(y) as linear of Xs
- View II: model Y as Bernoulli with p(y=1|x) as p(Head)
- View III: S" shape function compress to [0,1]
- View IV: models a linear classification boundary!
- View V: Two stages: summation + sigmoid

Today: Basic Neural Network Models

Data: X

Task: y

Representation: : x, f()

Score Function: L()

Search/Optimization : argmin()

Models, Parameters, metrics

X: Tabular / 2D / 1D / 3D / Graph /...

Classification / Regression

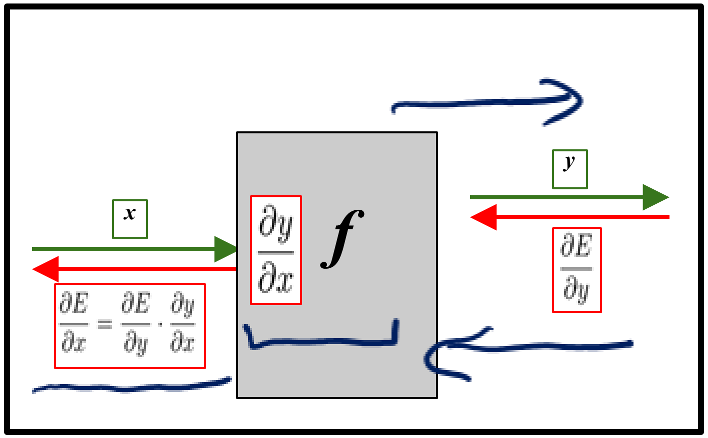
Multilayer Network topology

neg Log-likelihood , Cross-Entropy / MSE / Many more

SGD / Backprop

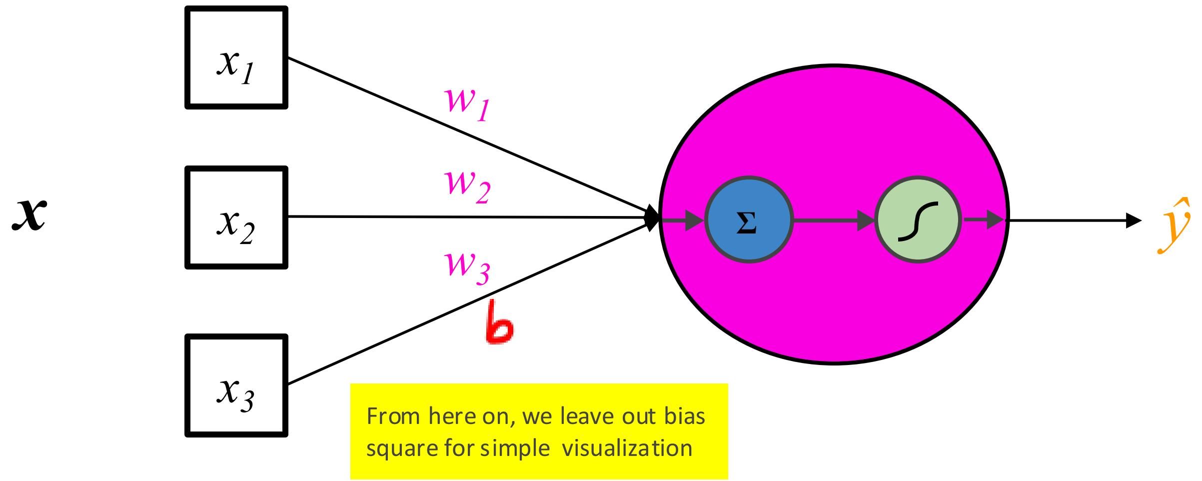
Weights and biases in NN Layers / Accuracy / F1

Building Deep Neural Nets



http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

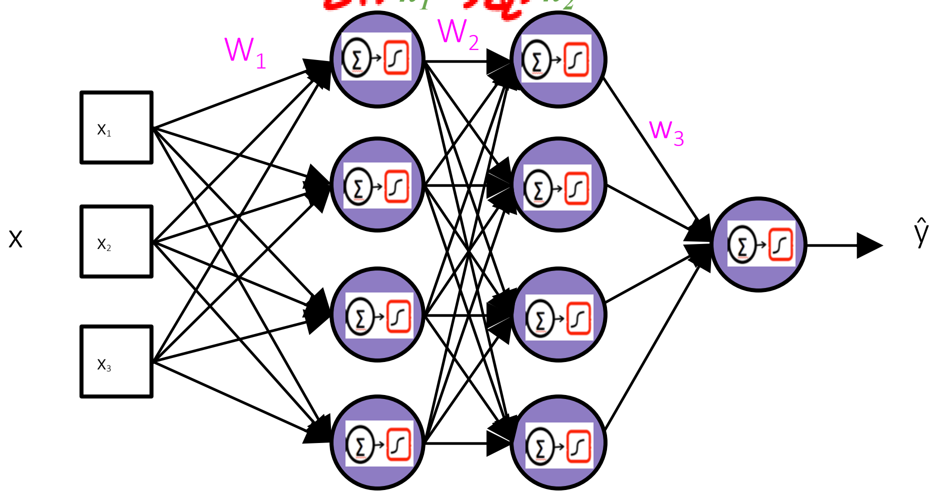
Neuron Representation



The linear transformation and nonlinearity together is typically considered a single neuron

Multi-Layer Perceptron (MLP)- (Feed-Forward NN)

$z_1 \to h_1 \to z_2 \to h_2$

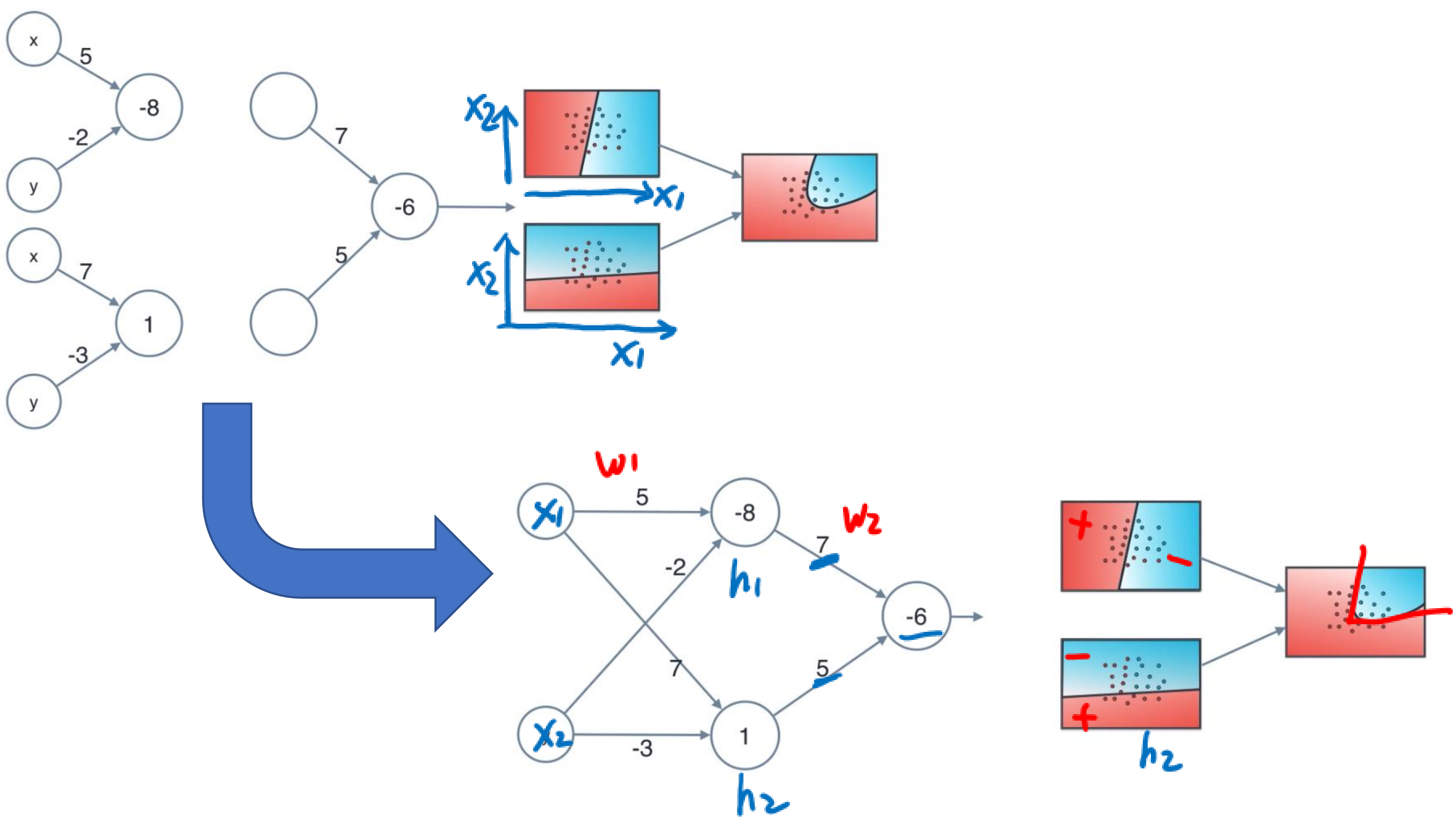


hidden layer 1 output → $h_1 = sigmoid(z_1)$

hidden layer 2 output → $h_2 = sigmoid(z_2)$

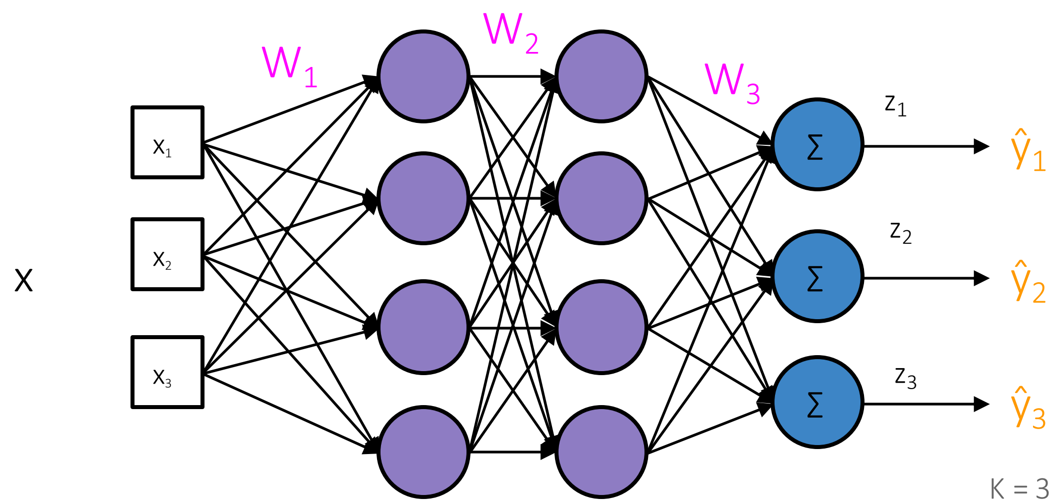
$$
\begin{aligned}
z_1 &= W_1^T x \\
h_1 &= sigmoid(z_1) \\
z_2 &= W_2^T h_1 \\
h_2 &= sigmoid(z_2) \\
z_3 &= w_3^T h_2 \\
\hat{y} &= sigmoid(z_3)
\end{aligned}
$$

$X \xrightarrow[w_1]{} z_1 \longrightarrow h_1 \xrightarrow[w_2]{} z_2 \longrightarrow h_2 \xrightarrow[w_3]{} \hat{y}$


x
5
-8
-2
y
x
7
1
-3
y
7
-6
5
X2
X1
X2
X1
W1
X1
5
-8
W2
7
-2
h1
-6
7
5
X2
-3
1
h2
+
-
-
+
h2

Recap: Multi-Class Classification Loss

Cross Entropy Loss



$$\hat{y}_i = \frac{e^{z_i}}{\sum_j e^{z_j}} = P(\, y_i = 1 \mid x \,)$$

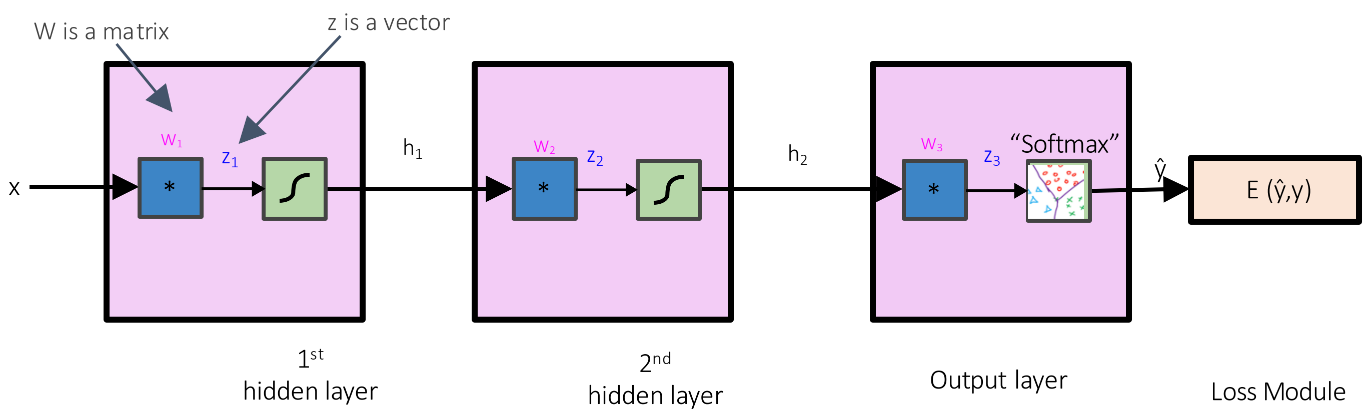
“Softmax” function. Normalizing function which converts each class output to a probability.

$$\underset{y}{\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}} \quad \underset{\hat{y}}{\begin{pmatrix} 0.1 \\ 0.7 \\ 0.2 \end{pmatrix}}$$

$$\boxed{E = \text{loss} = -\sum_{j = 1 \ldots K} y_j \log \hat{y}_j}$$

“0” for all except true class

e.g., “Block View” of multi-layered multi-class NN



$$x \xrightarrow{W_1} h_1 \xrightarrow{W_2} h_2 \xrightarrow{W_3} \hat{y} \longrightarrow E(\hat{y}, y)$$

Extra

argmin_w $\{ f_4 (f_3 (f_2 (f_1 (\,)))) \}$

f_1	$z_1 = x_1 w_1 + x_2 w_3 + b_1$ $z_2 = x_1 w_2 + x_2 w_4 + b_2$
f_2	$h_1 = \frac{exp(z_1)}{1 + exp(z_1)}$ $h_2 = \frac{exp(z_2)}{1 + exp(z_2)}$
f_3	$\hat{y} = h_1 w_5 + h_2 w_6 + b_3$
f_4	$E = (y - \hat{y})^2$

Input	Output	Local Gradients=∂Output /∂Input
$x_1, x_2, w_1, \ldots$	z_1, z_2	
z_1, z_2	h_1, h_2	$\frac{\partial h_1}{\partial z_1} = h_1(1-h_1)$
w_5, h_1, h_2	$\hat{y}$	$\partial \hat{y} / \partial h_1 = w_5$
$\hat{y}$	loss E	$\partial E / \partial \hat{y} = -2(y-\hat{y})$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial w_1} = \frac{\partial f_4}{\partial f_3} \frac{\partial f_3}{\partial f_2} \frac{\partial f_2}{\partial f_1} \frac{\partial f_1}{\partial w_1}$$

$$= -2(y-\hat{y}) \frac{\partial (h_1 w_5 + h_2 w_6 + b_3)}{\partial w_1}$$

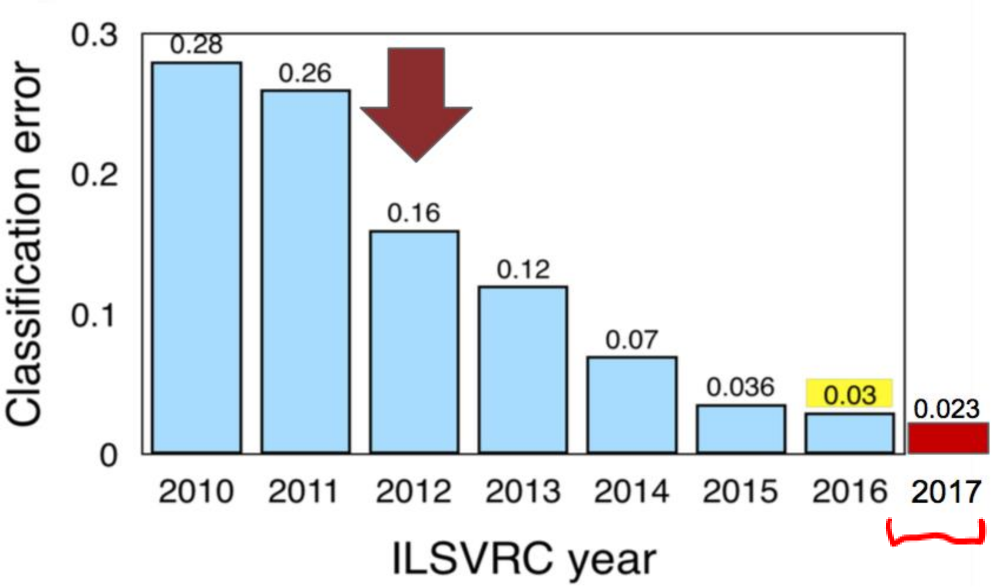
$$= -2(y-\hat{y}) \left(w_5 \frac{\partial h_1}{\partial w_1} + w_6 \frac{\partial h_2}{\partial w_1} \right)$$

$$= -2(y-\hat{y}) w_5 \frac{\partial h_1}{\partial w_1} = -2(y-\hat{y}) w_5 \frac{\partial h_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}$$

$$= \underbrace{-2(y-\hat{y})}_{f_4 \text{ local}} \underbrace{w_5}_{f_3 \text{ local}} \underbrace{h_1(1-h_1)}_{f_2 \text{ local}} \underbrace{x_1}_{\frac{\partial f_1}{\partial w_1}}$$

ImageNet Challenge

- 2010-11: hand-crafted computer vision pipelines
- 2012-2016: ConvNets
  - 2012: AlexNet
    - major deep learning success
  - 2013: ZFNet
    - improvements over AlexNet
  - 2014
    - VGGNet: deeper, simpler
    - InceptionNet: deeper, faster
  - 2015
    - ResNet: even deeper
  - 2016
    - ensembled networks
  - 2017
    - Squeeze and Excitation Network



Adapt from From NIPS 2017 DL Trend Tutorial

Metric	Formula	Interpretation
Accuracy	$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$	Overall performance of model
Precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Coverage of actual positive sample
Specificity	$\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$	Coverage of actual negative sample
F1 score	$\frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$	ed classes

	actual	
	$+$	$-$
predicted$+$	TP	FP
predicted$-$	FN	TN

Today: Convolutional Network Models on 2D Grid / Image

Data: X

↓

Task: y

↓

Representation: : x, f()

↓

Score Function: L()

↓

Search/Optimization : argmin()

↓

Models, Parameters, metrics

X: 2D Grid Imaging

↓

Classification

↓

Convolutional Neural Networks

↓

Cross Entropy Loss

↓

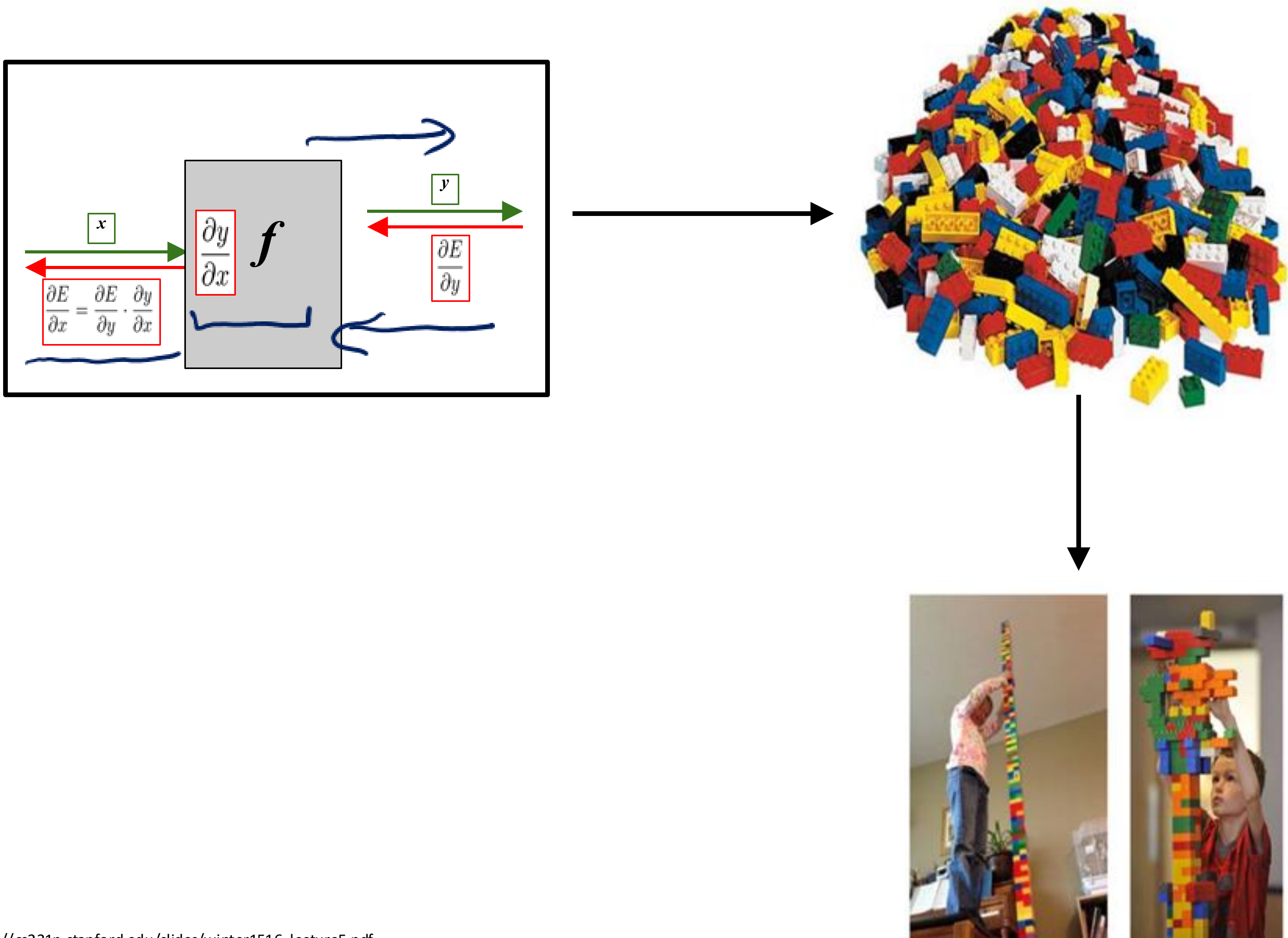
SGD / Backprop

↓

weights, bias / architecture /

/ Accuracy / F1

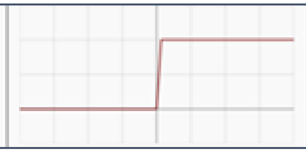
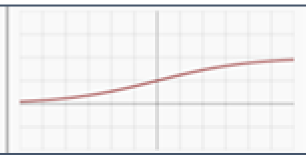
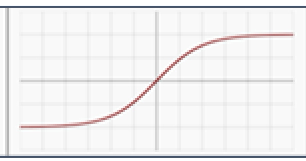
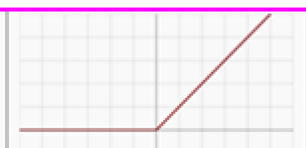
Building Deep Neural Nets



http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

E.g., Many Possible Nonlinearity Functions

(aka transfer or activation functions)

Name	Plot	Equation	Derivative (w.r.t x)
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Rectifier (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

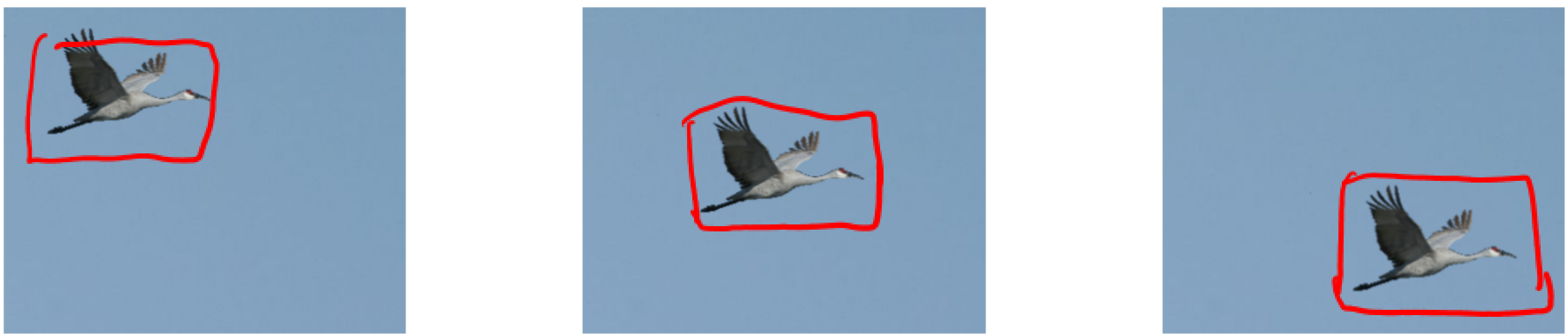
usually works best in practice

https://en.wikipedia.org/wiki/Activation_function#Comparison_of_activation_functions

CNN models Locality and Translation Invariance

Important Block: Convolutional Neural Networks (CNN)

- Prof. Yann LeCun invented CNN in 1998
- First NN successfully trained with many layers



The bird occupies a local area and looks the same in different parts of an image.
We should construct neural nets which exploit these properties!

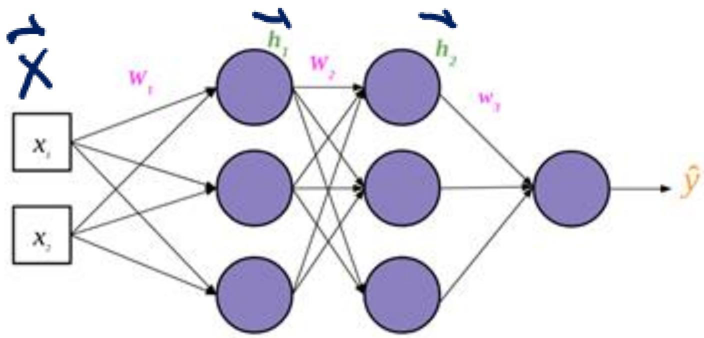
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Adapt from From NIPS 2017 DL Trend Tutorial

TF Keras Sample Code

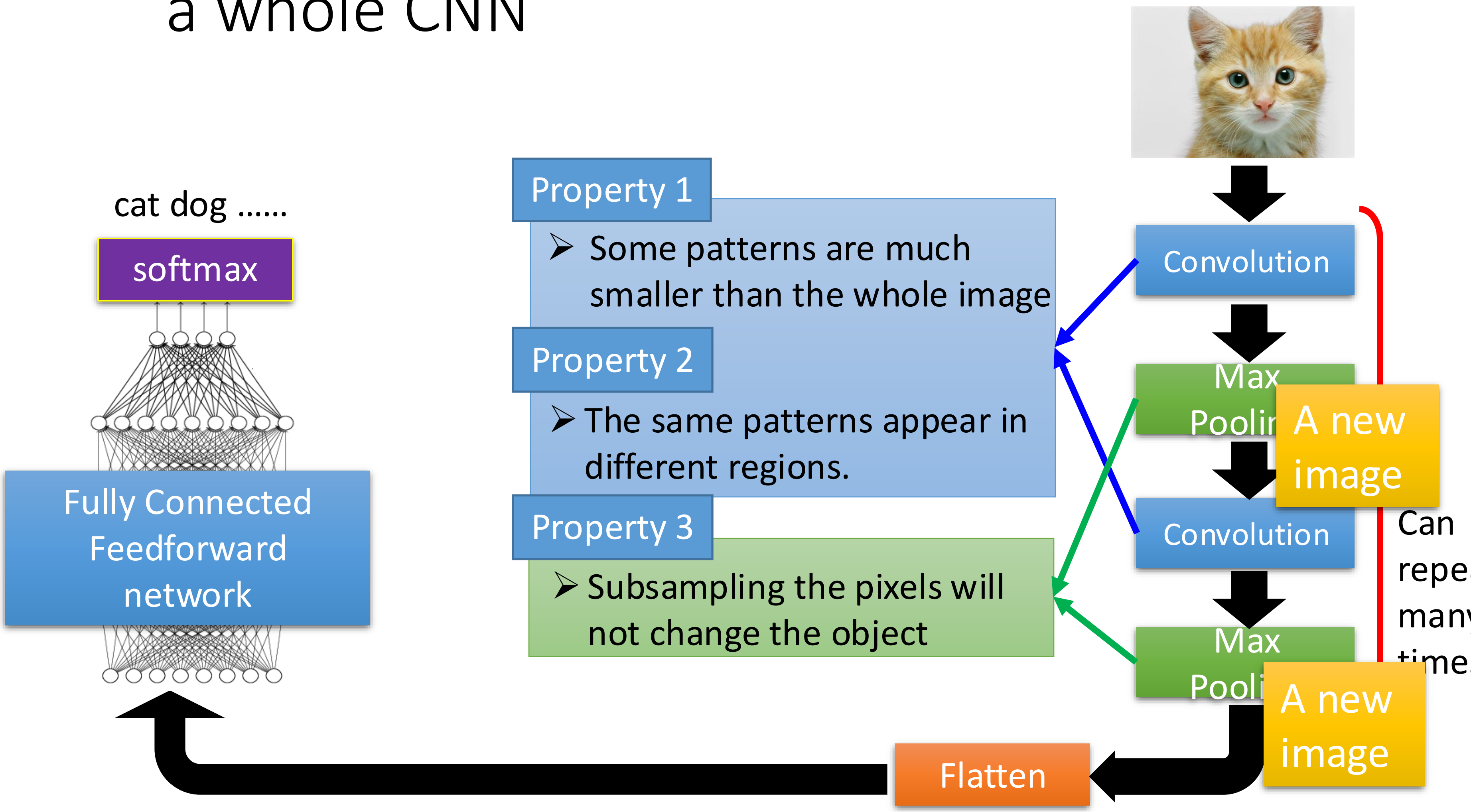
https://www.kaggle.com/code/shawon10/covid-19-diagnosis-from-images-using-densenet121

Pytorch Sample Code



```
import torch.nn as nn
import torch.nn.functional as F

class ThreeLayerNet(torch.nn.Module):
    def __init__(self, d_in, d_hidden, d_out):
        super().__init__()
        self.W1 = nn.Linear(d_in,d_hidden)
        self.W2 = nn.Linear(d_hidden,d_hidden)
        self.w3 = nn.Linear(d_hidden,d_out)
        self.nonlinear = nn.Sigmoid()

    def forward(self, x):
        h1 = self.nonlinear(self.W1(x))
        h2 = self.nonlinear(self.W2(h1))
        y_hat = self.nonlinear(self.w3(h2))
        return y_hat

model = ThreeLayerNet(2,3,1)
```

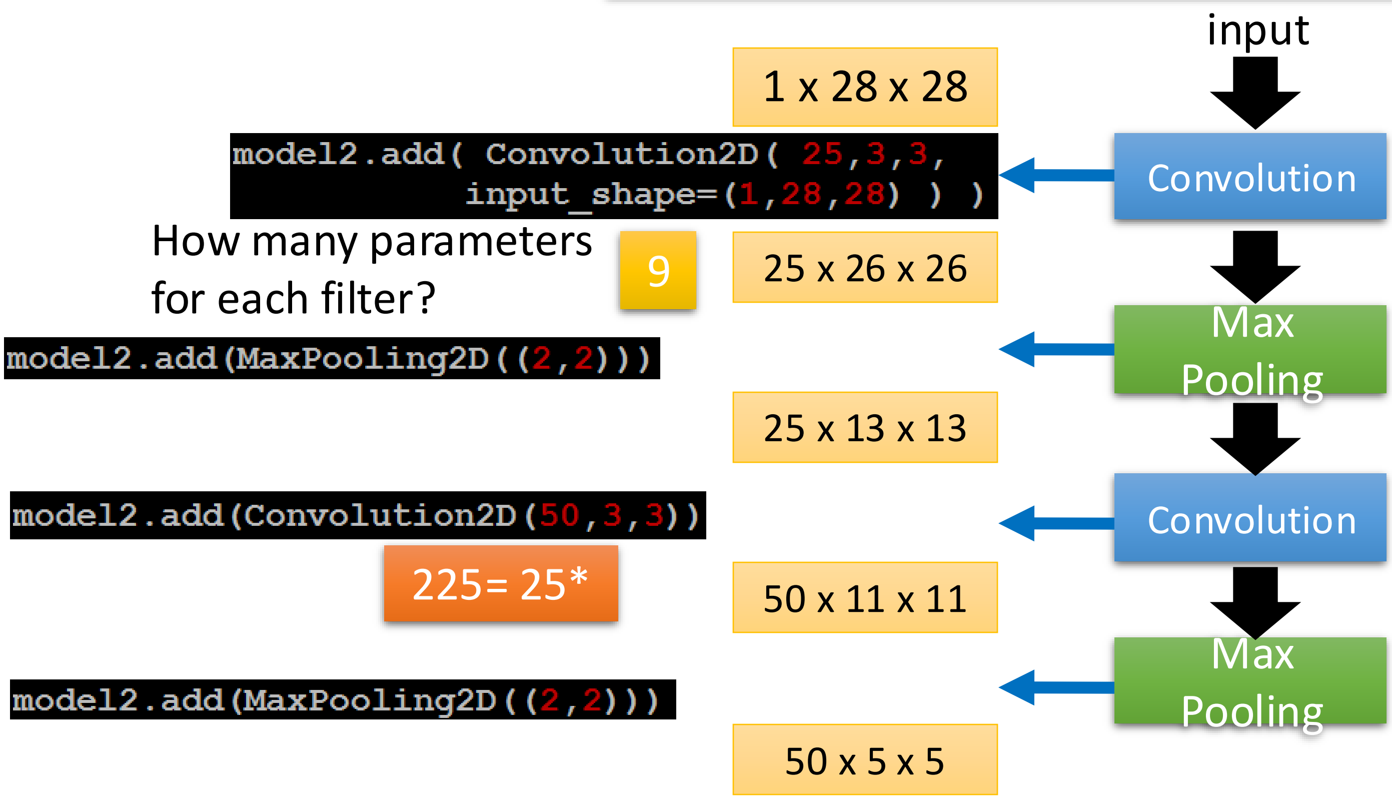
a whole CNN



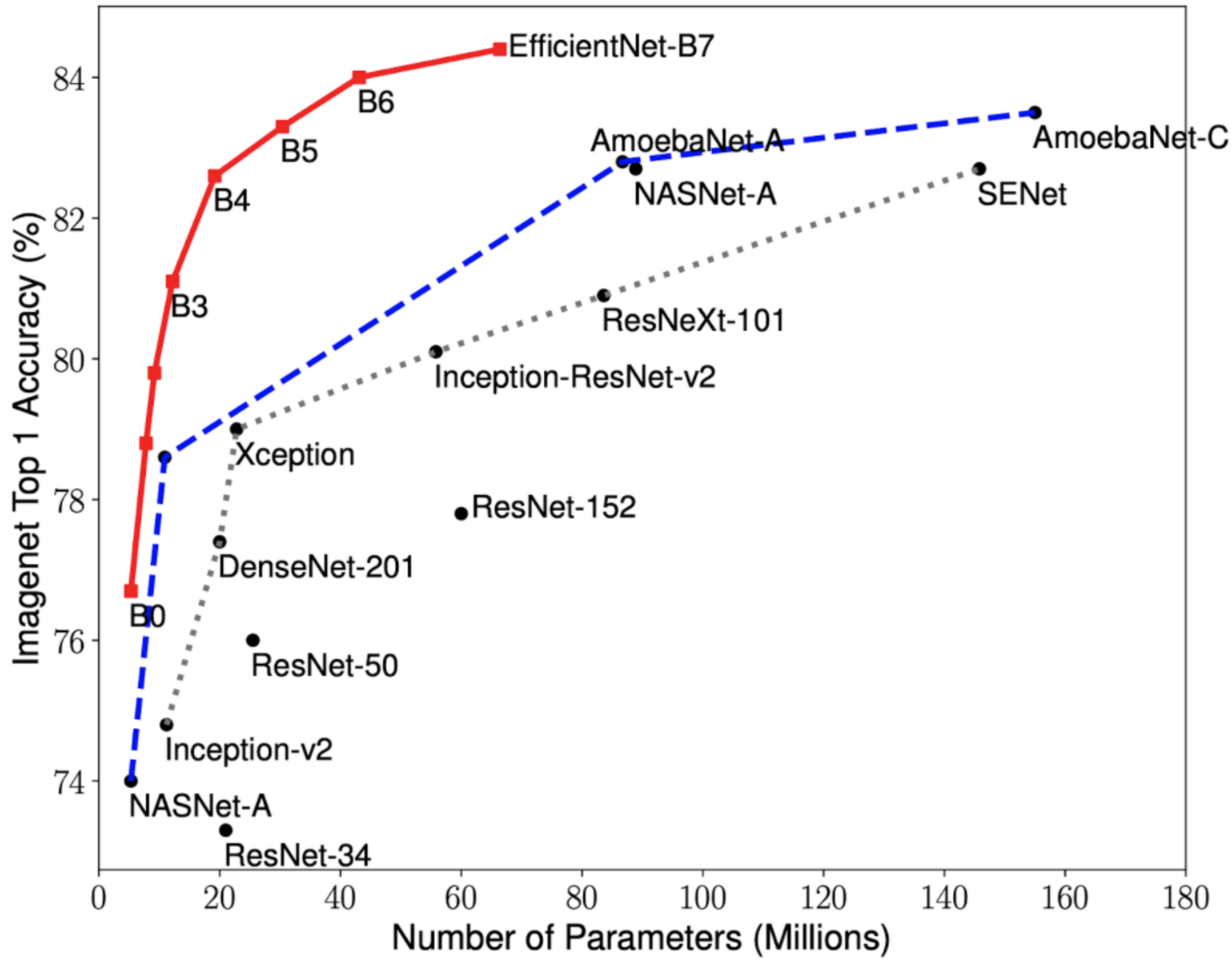
Adapted Dr. Hung-yi Lee’s CNN slides

CNN in Keras

network structure and ***input format***
(vector -> 3-D tensor)



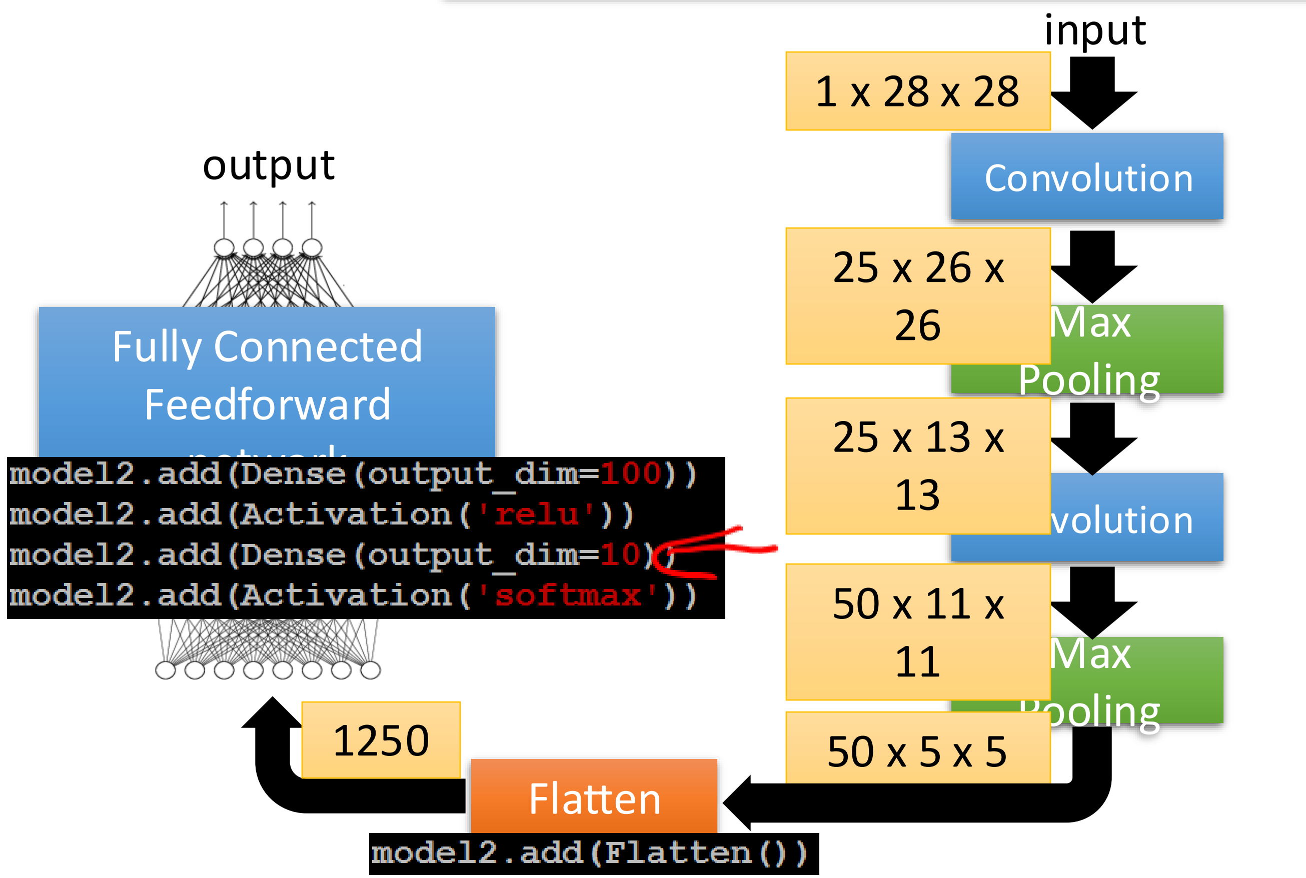
Dr. Hung-yi Lee’s CNN slides



https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***



Dr. Hung-yi Lee's CNN slides

Lecture 14: Dimension Reduction

Today: Dimensionality Reduction (Two Ways)

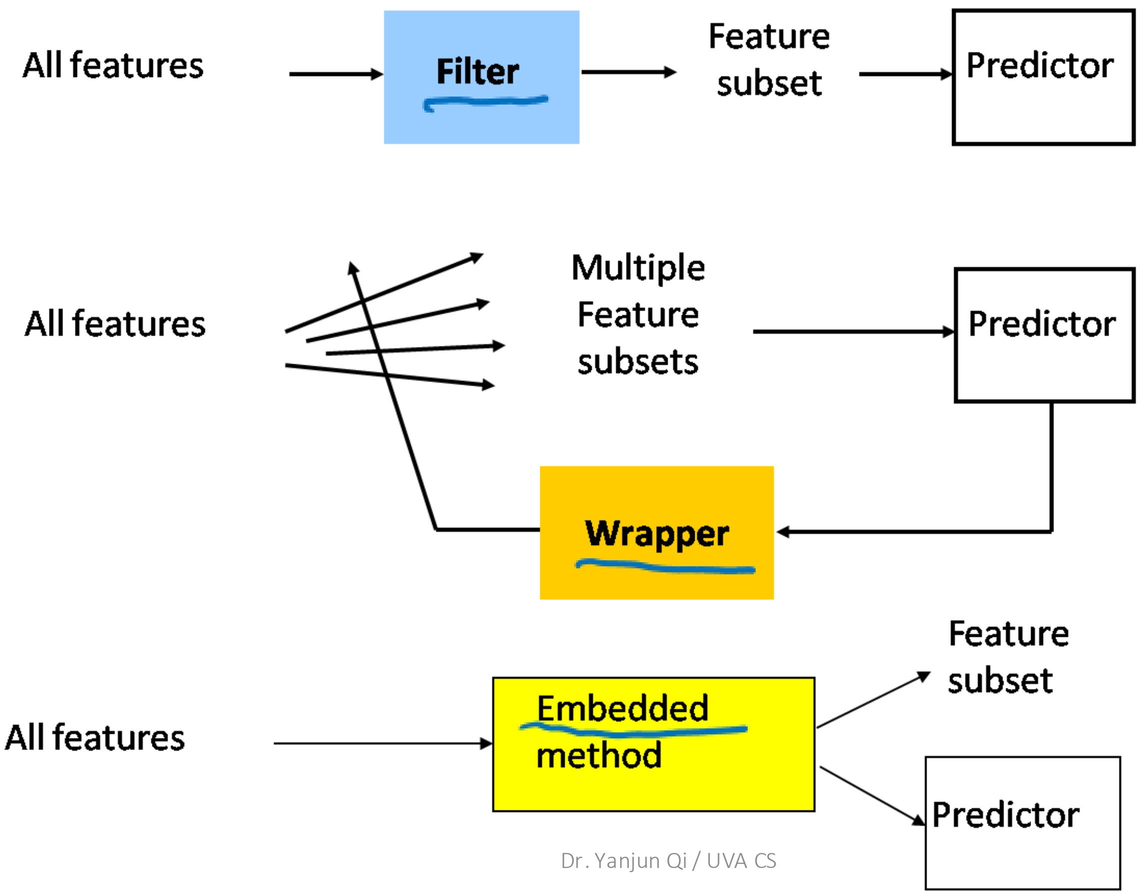
Feature selection: chooses a subset of the original features.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \longrightarrow \mathbf{x}' = \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{kK} \end{bmatrix}$$

K<<N

Summary: Feature Selection
=> filters vs. wrappers vs. embedding

- Main goal: rank subsets of useful features



From Dr. **Isabelle Guyon**

(I) Filtering : (many choices)

Method		X			Y			Comments
Name	Formula	B	M	C	B	M	C	
Bayesian accuracy	Eq. 3.1	+	s		+	s		Theoretically the golden standard, rescaled Bayesian relevance Eq. 3.2.
Balanced accuracy	Eq. 3.4	+	s		+	s		Average of sensitivity and specificity; used for unbalanced dataset, same as AUC for binary targets.
Bi-normal separation	Eq. 3.5	+	s		+	s		Used in information retrieval.
F-measure	Eq. 3.7	+	s		+	s		Harmonic of recall and precision, popular in information retrieval.
Odds ratio	Eq. 3.6	+	s		+	s		Popular in information retrieval.
Means separation	Eq. 3.10	+	i	+	+			Based on two class means, related to Fisher's criterion.
T-statistics	Eq. 3.11	+	i	+	+			Based also on the means separation.
Pearson correlation	Eq. 3.9	+	i	+	+	i	+	Linear correlation, significance test Eq. 3.12, or a permutation test.
Group correlation	Eq. 3.13	+	i	+	+	i	+	Pearson's coefficient for subset of features.
χ^2	Eq. 3.8	+	s		+	s		Results depend on the number of samples m.
Relief	Eq. 3.15	+	s	+	+	s	+	Family of methods, the formula is for a simplified version ReliefX, captures local correlations and feature interactions.
Separability Split Value	Eq. 3.41	+	s	+	+	s		Decision tree index.
Kolmogorov distance	Eq. 3.16	+	s	+	+	s	+	Difference between joint and product probabilities.
Bayesian measure	Eq. 3.16	+	s	+	+	s	+	Same as Vajda entropy Eq. 3.23 and Gini Eq. 3.39.
Kullback-Leibler divergence	Eq. 3.20	+	s	+	+	s	+	Equivalent to mutual information.
Jeffreys-Matusita distance	Eq. 3.22	+	s	+	+	s	+	Rarely used but worth trying.
Value Difference Metric	Eq. 3.22	+	s		+	s		Used for symbolic data in similarity-based methods, and symbolic feature-feature correlations.
Mutual Information	Eq. 3.29	+	s	+	+	s	+	Equivalent to information gain Eq. 3.30.
Information Gain Ratio	Eq. 3.32	+	s	+	+	s	+	Information gain divided by feature entropy, stable evaluation.
Symmetrical Uncertainty	Eq. 3.35	+	s	+	+	s	+	Low bias for multivalued features.
J-measure	Eq. 3.36	+	s	+	+	s	+	Measures information provided by a logical rule.
Weight of evidence	Eq. 3.37	+	s	+	+	s	+	So far rarely used.
MDL	Eq. 3.38	+	s		+	s		Low bias for multivalued features.

Guyon-Elisseeff, JMLR 2004; Springer 2006

(2) Wrapper : Feature Subset Selection

Wrapper Methods

- Learner is considered a black-box
- Interface of the black-box is used to score subsets of variables according to the predictive power of the learner when using the subsets.

- Results vary for different learners

(b). Search: even more search strategies for selecting feature subset

$p \longrightarrow 2^p$ feature subsets

- **Forward selection** or **backward elimination.**
- **Beam search:** keep k best path at each step.
- **GSFS:** generalized sequential forward selection – when (n-k) features are left try all subsets of g features. More trainings at each step, but fewer steps.
- **PTA(l,r):** plus l , take away r – at each step, run SFS l times then SBS r times.
- **Floating search**: One step of SFS (resp. SBS), then SBS (resp. SFS) as long as we find better subsets than those of the same size obtained so far.

From Dr. **Isabelle Guyon**

(3) Embedded

- Embedding approach:

uses a predictor to build a (single) model with a subset of features that are internally selected.

lasso

elastiNet

Today: Dimensionality Reduction (Two Ways)

Feature extraction: finds a set of new features (i.e., through some mapping f()) from the existing features.

Feature selection: chooses a subset of the original features.

The mapping f() could be linear or non-linear

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \xrightarrow{f()} \mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_K \end{bmatrix}$$

K<<N

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \longrightarrow \mathbf{x}' = \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{kK} \end{bmatrix}$$

K<<N

Feature Extraction (linear or nonlinear)

- Linear combinations are particularly attractive because they are simpler to compute and analytically tractable.

- Given $x \in R^p$, find an N x K matrix U such that:

$$y = U^T x \in R^K \text{ where } K<P$$

This is a projection from the N-dimensional space to a K-dimensional space.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \xrightarrow{U^T} \mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_K \end{bmatrix}$$

Feature Extraction (cont'd)

- Commonly used linear feature extraction methods:
  - Principal Components Analysis (PCA): Seeks a projection that **preserves** as much **information** in the data as possible.
  - Linear Discriminant Analysis (LDA): Seeks a projection that **best discriminates** the data.

- Recent nonlinear feature extraction methods:
  - Like Word Embedding / Autoencoder / …

Principal Component Analysis

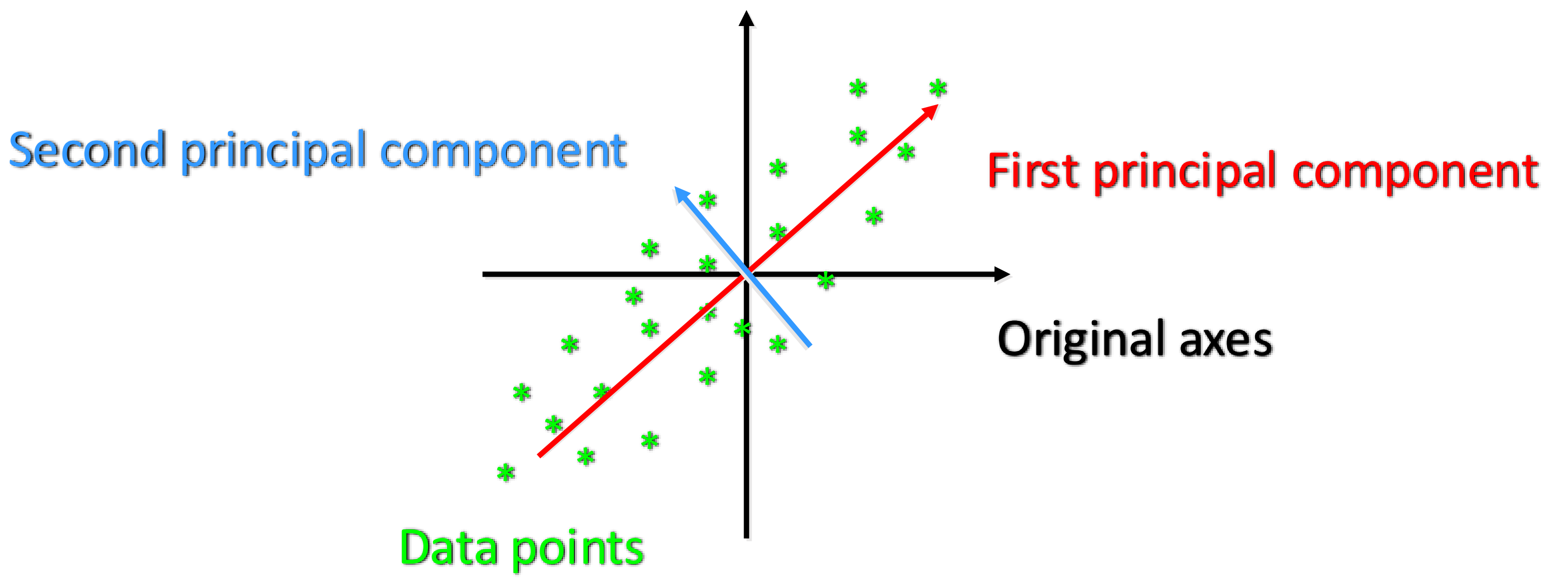
Task → Representation → Score Function → Search/Optimization → Models, Parameters

Dimension Reduction → Gaussian assumption → Direction of maximum variance → Eigen-decomp → Principal components

How does PCA work? Explaining Variance

- Each PC always explains some proportion of the total variance in the data. Between them they explain everything
  - PC1 always explains the most
  - PC2 is the next highest etc. etc.

$p \longrightarrow PCs$



Auto Encoder

Data

Task

Representation

Score Function

Search/Optimization

Models,
Parameters

X: Tabular / 2D / 1D / 3D / Graph /…

Dimension Reduction /
Representations Learning

Gaussian assumption

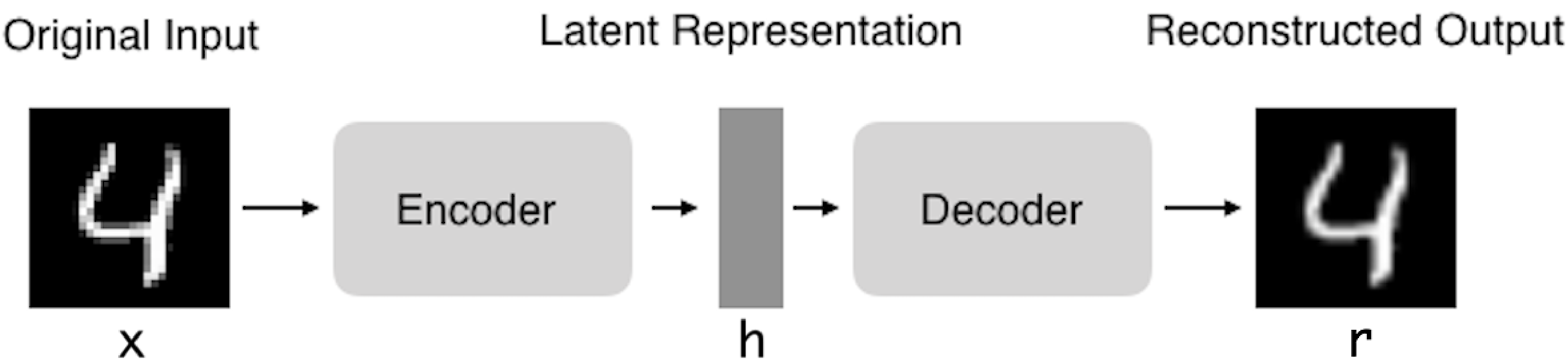
Reconstruction Error /
Variational MLE

SGD

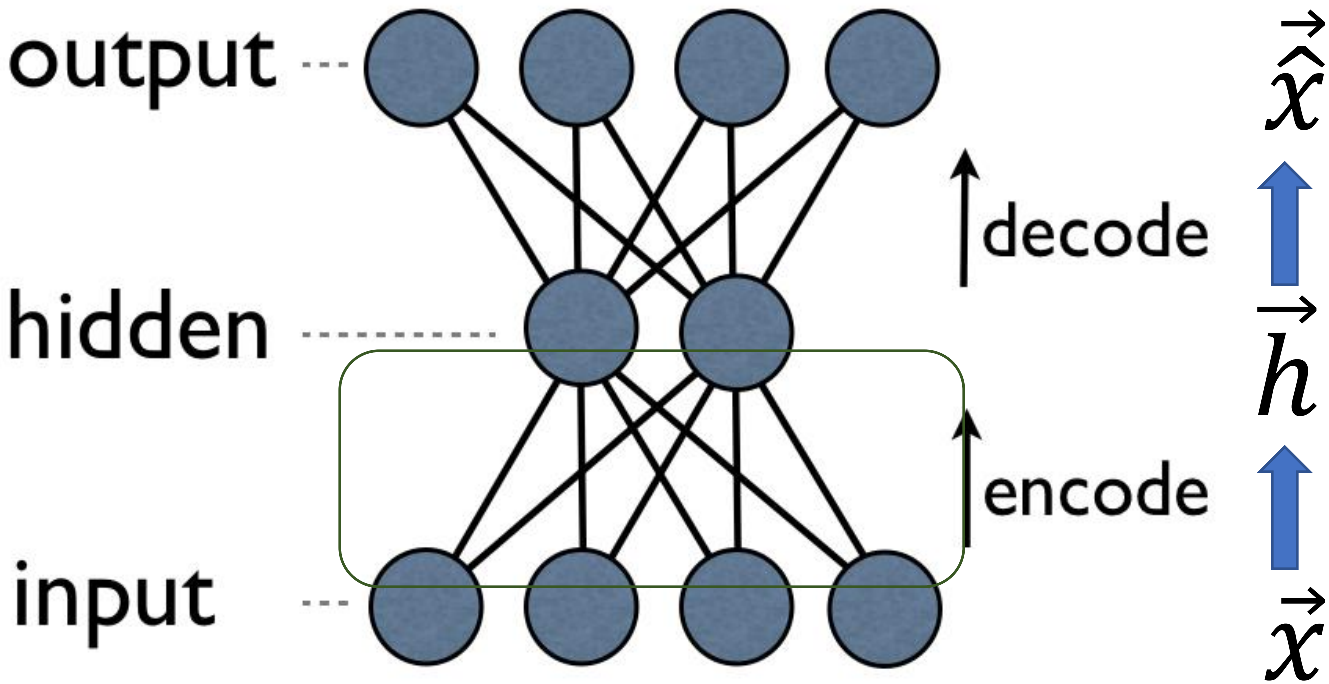
Disentangled Representations

Autoencoders: structure

- Encoder: compress input into a latent-space of usually smaller dimension. $h = f(x)$
- Decoder: reconstruct input from the latent space. $r = g(f(x))$ with r as close to x as possible



https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f an auto-encoder-decoder is trained to reproduce the input



Minimize diff

$$|\vec{\hat{x}} - \vec{x}|$$

Reconstruction Loss: force the 'hidden layer' units to become good / reliable feature detectors

https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

10/28

Today: Neural Network Models on 1D Grid / Language Data

Data: X

Task: y

Representation: : x, f()

Score Function: L()

Search/Optimization : argmin()

Models, Parameters, metrics

X: 1D Grid Text

Y: Many possible Kinds,

e.g., 1D Grid Text

Word2Vec/ Recurrent / Transformer

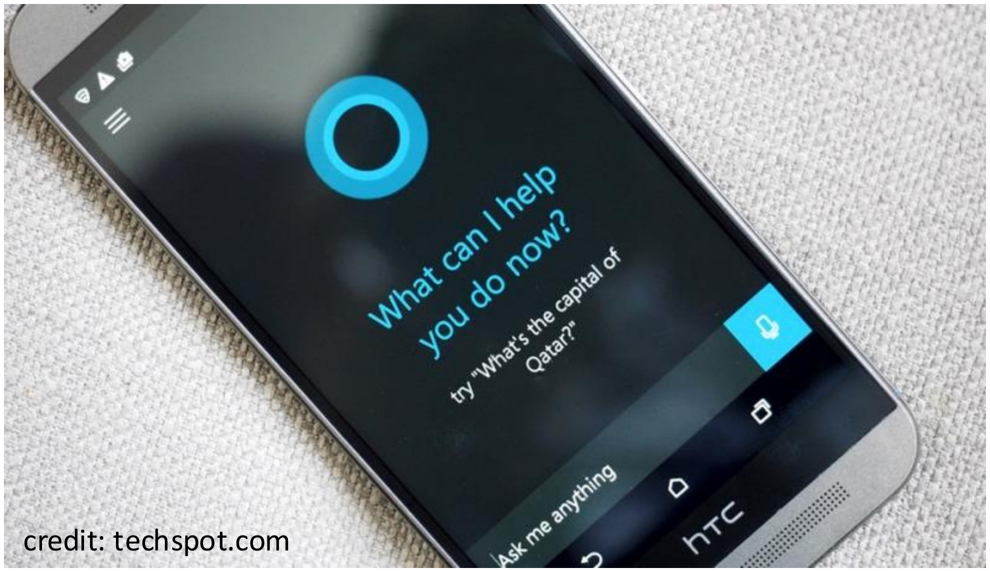
Many self-supervised kinds, e.g., skip-gram / mask LM

SGD / Backprop

weights, bias / hyperparameters / Accuracy / F1 / more metris

Deep Learning for natural language processing, e.g.,

Digital personal assistant



credit: techspot.com

- Semantic parsing – understand tasks
- Entity linking – “my wife” = “Kellie” in the phone book

Credit: kaiwai Chang

f() on natural language

- Before Deep NLP (Pre 2012)
  - (BOW / LSI / Topic LDA)
- Word2Vec (2013-2016)
  - (GloVe/ FastText)
- Recurrent NN (2014-2016)
  - LSTM
  - Seq2Seq
- Attention / Self-Attention (2016 – now)
  - Attention
  - Transformer (self-attention, attention only)
  - BERT / XLNet/ GPT-x / T5 ...

Recap: The bag of words representation

$f\Bigg($

great	2
love	2
recommend	1
laugh	1
happy	1
...	...

$\Bigg)=c$

How to Represent A Word in DNN: Feature Extraction / Embedding

Solution:

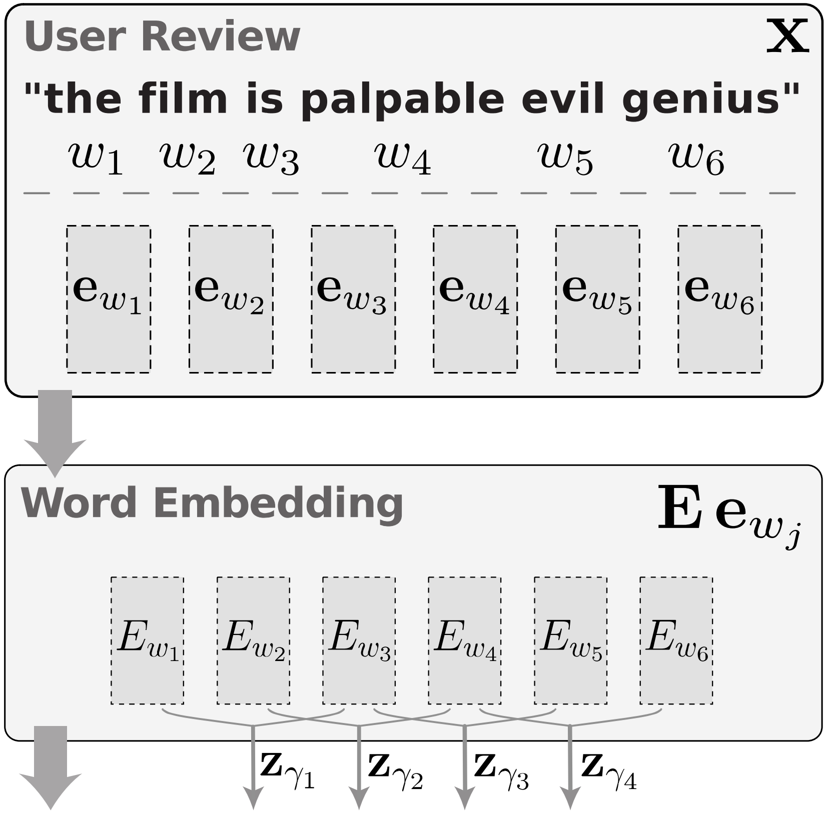
Distributional Word Embedding Vectors

- GloVe (Global Vectors)
  - Pennington et al., 2014
  - Skip-gram
- fasttext
  - Bojanowski et al., 2017

However, Natural language is

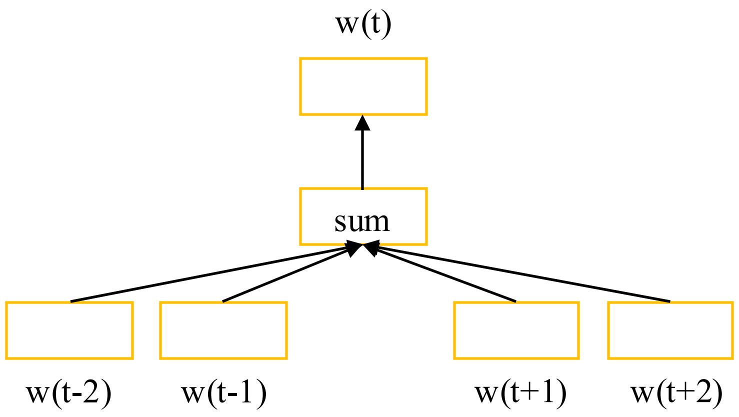
- Variable-length
- Composition of multiple words
- Word meaning is contextual

- Elmo
  - Peters, 2018
- BERT
  - Devlin et al., 2018

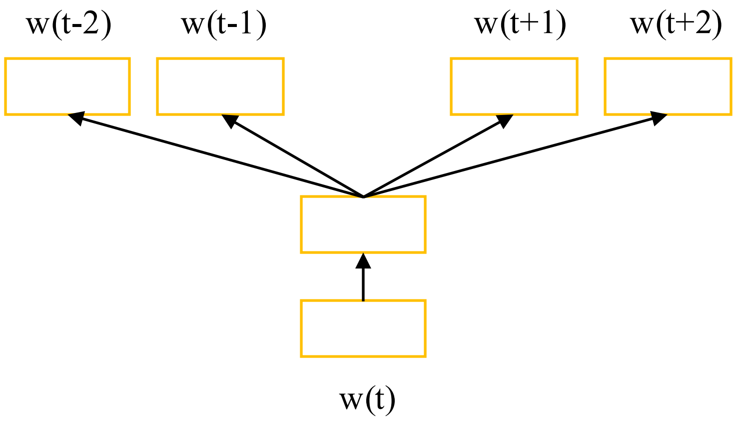


Word2vec: CBOW / SkipGram (Basic Word2Vec)

- Distributed representations of words and phrases and their compositionality (NIPS 2013, Mikolov et al.)
- CBOW
  - predict the input tokens based on context tokens
- SkipGram
  - predict context tokens based on input tokens



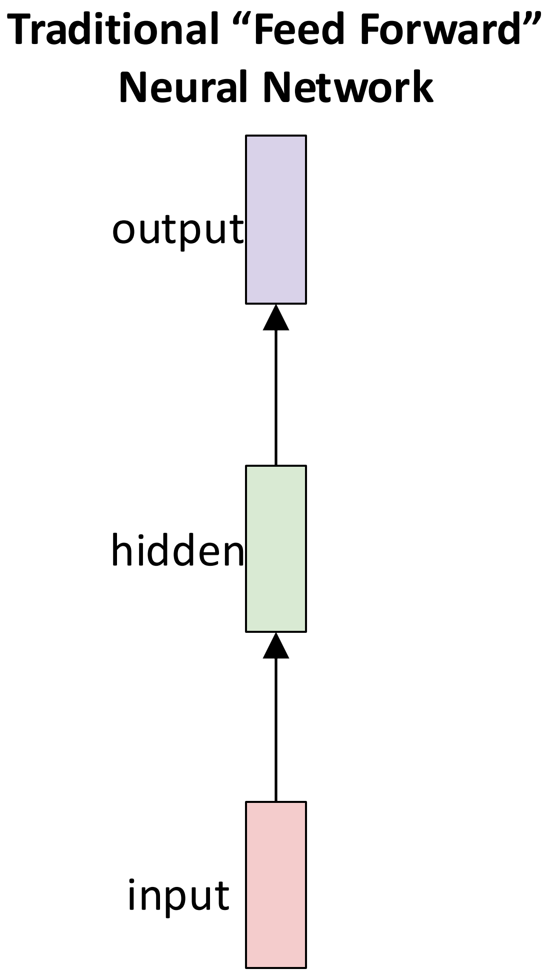
CBOW

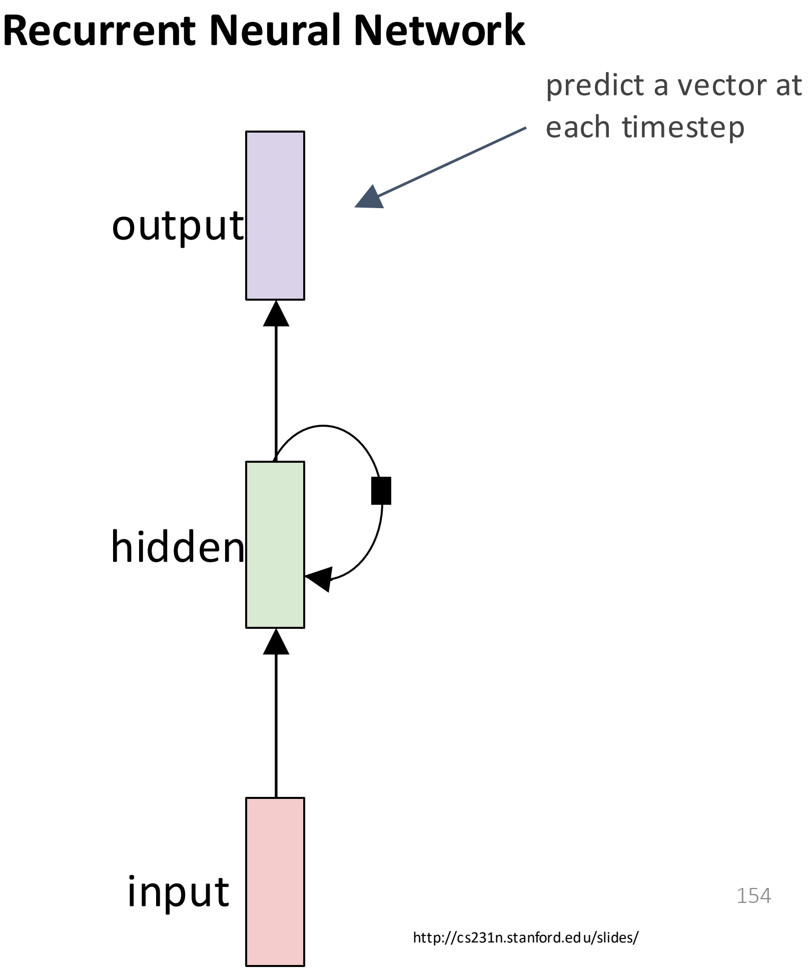


SkipGram

RNN models dynamic temporal dependency

- Make fully-connected layer model each unit recurrently
- Units form a directed chain graph along a sequence
- Each unit uses recent history and current input in modeling





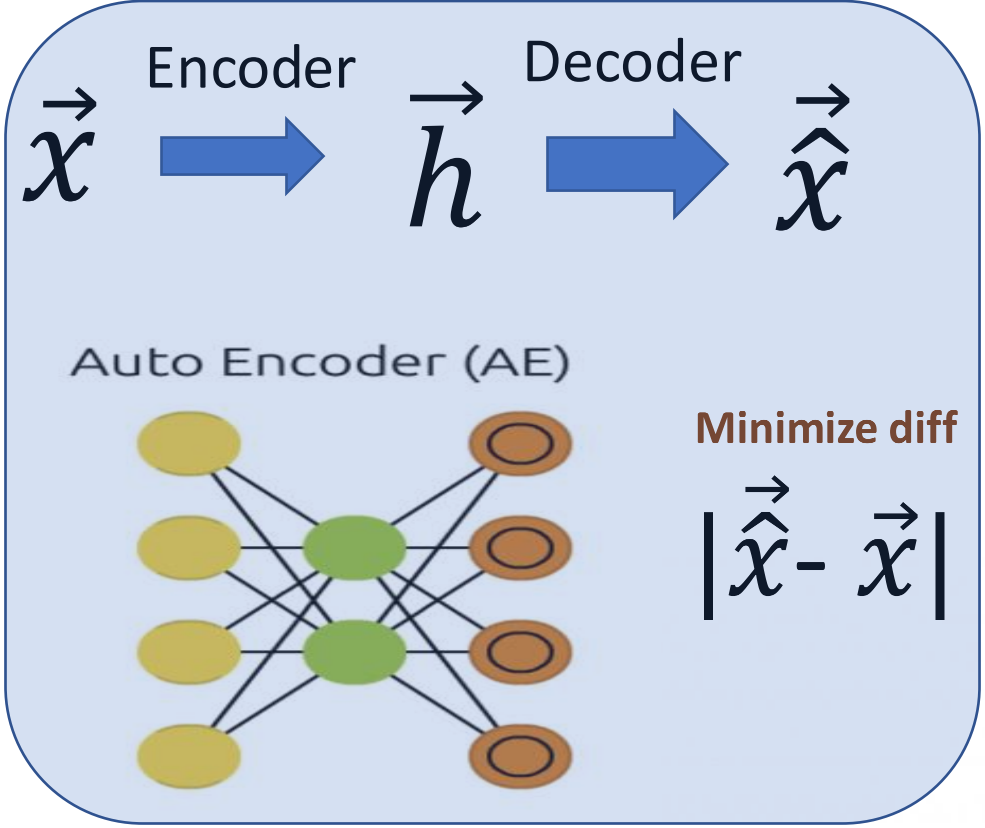
http://cs231n.stanford.edu/slides/

Seq2Seq for NLP Sequence-to-Sequence Generation Tasks

Given source sentences, learn an optimal model to automatically generate accurate and diversified target sentences that look like human generated sentences.

Source Sentence → Seq2Seq Model → Target Sentence

- Paraphrase generation: "How did Trump win the election?" → "How did Trump become president?"
- Dialogue generation: "You know French?" → "Sure do ... my Mom's from Canada"
- Question answering: "What was the name of the 1937 treaty?" → "Bald Eagle Protection Act"
- Style Transfer: "Just a dum funny question hahahaha" → "Just a senseless , funny question."



Seq2Seq

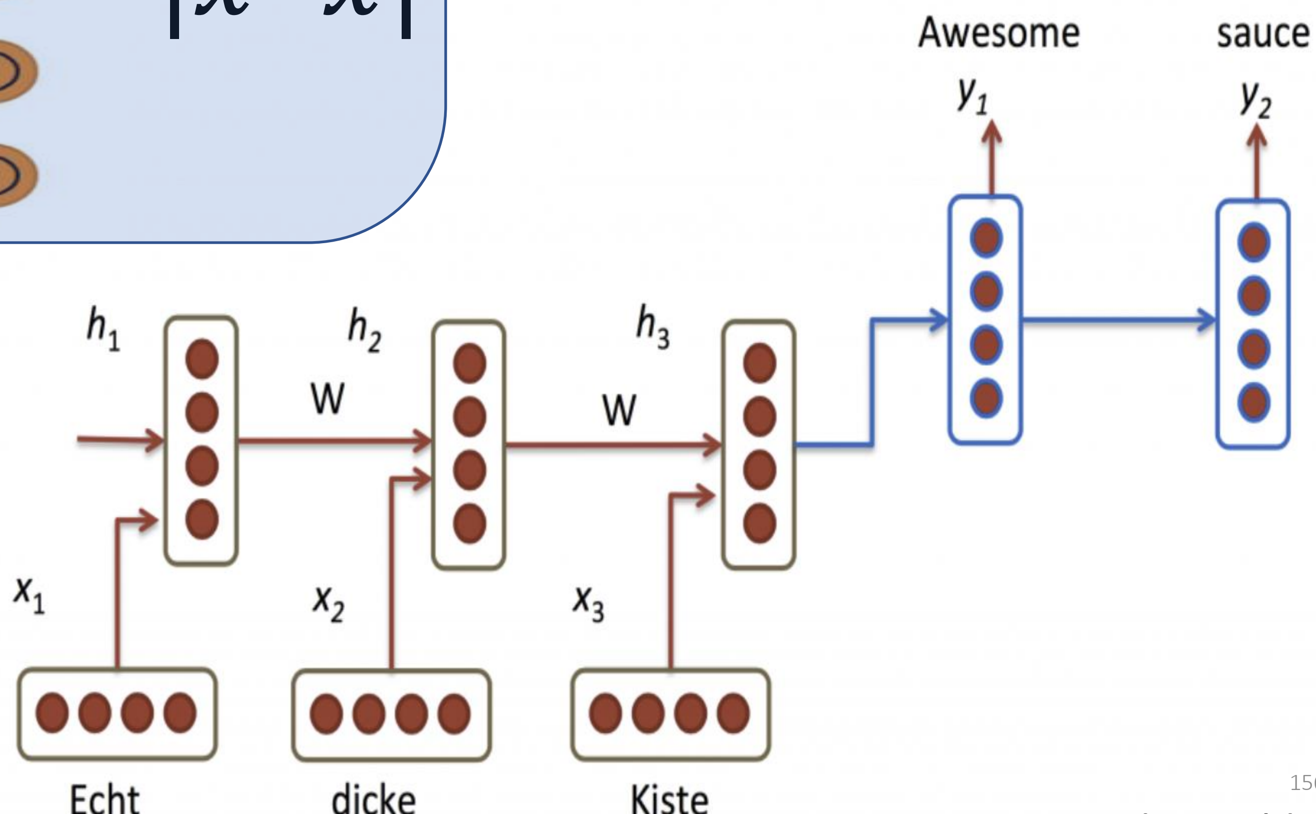
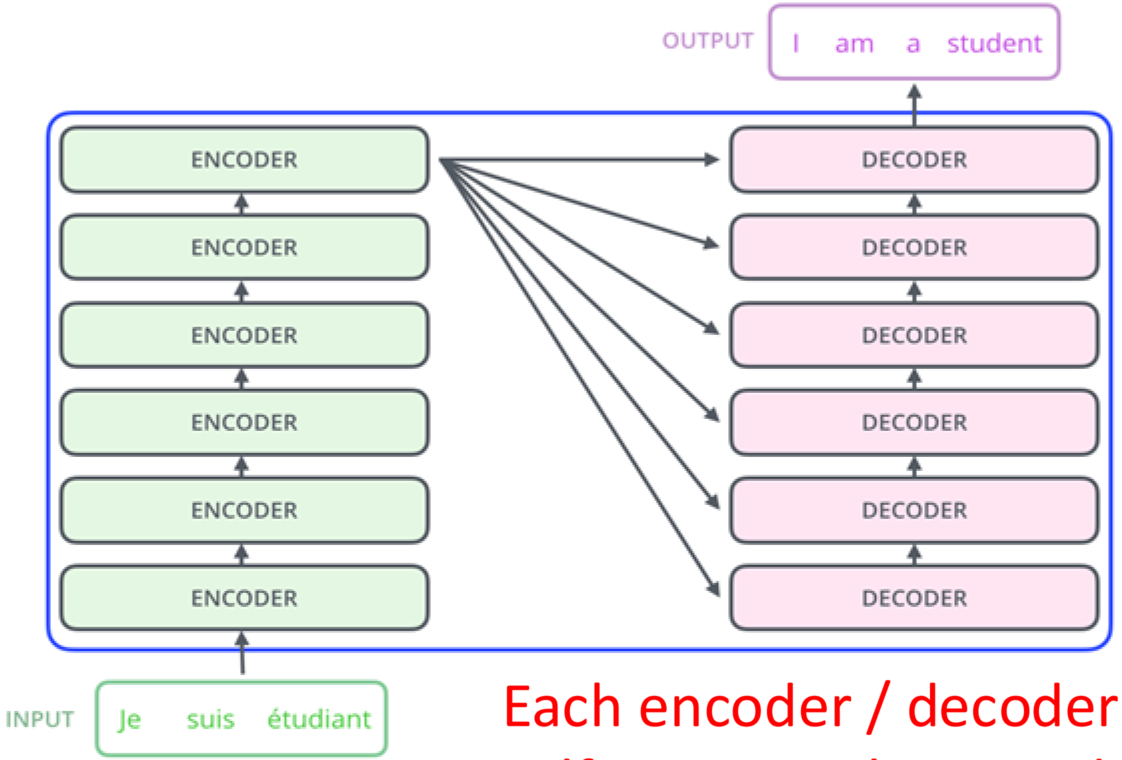
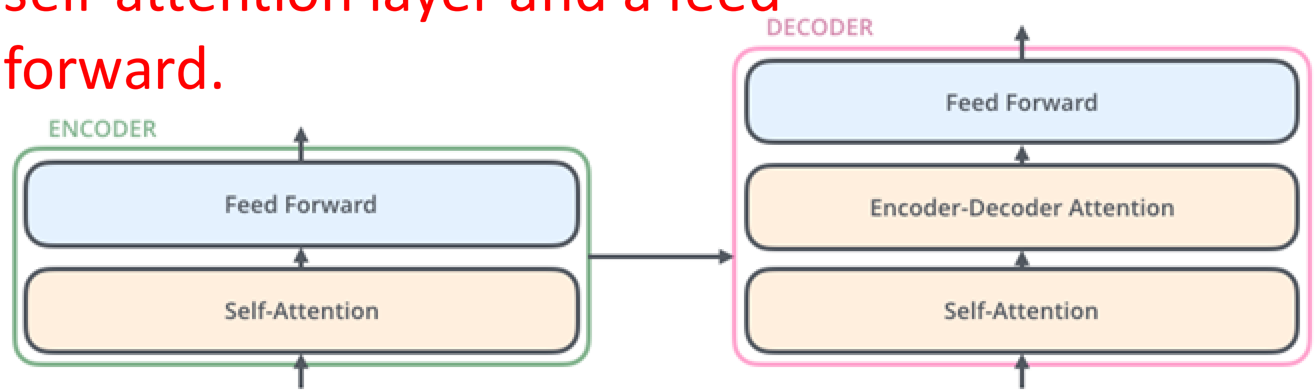


Image credit : wildML

Original Transformer is Seq2Seq model

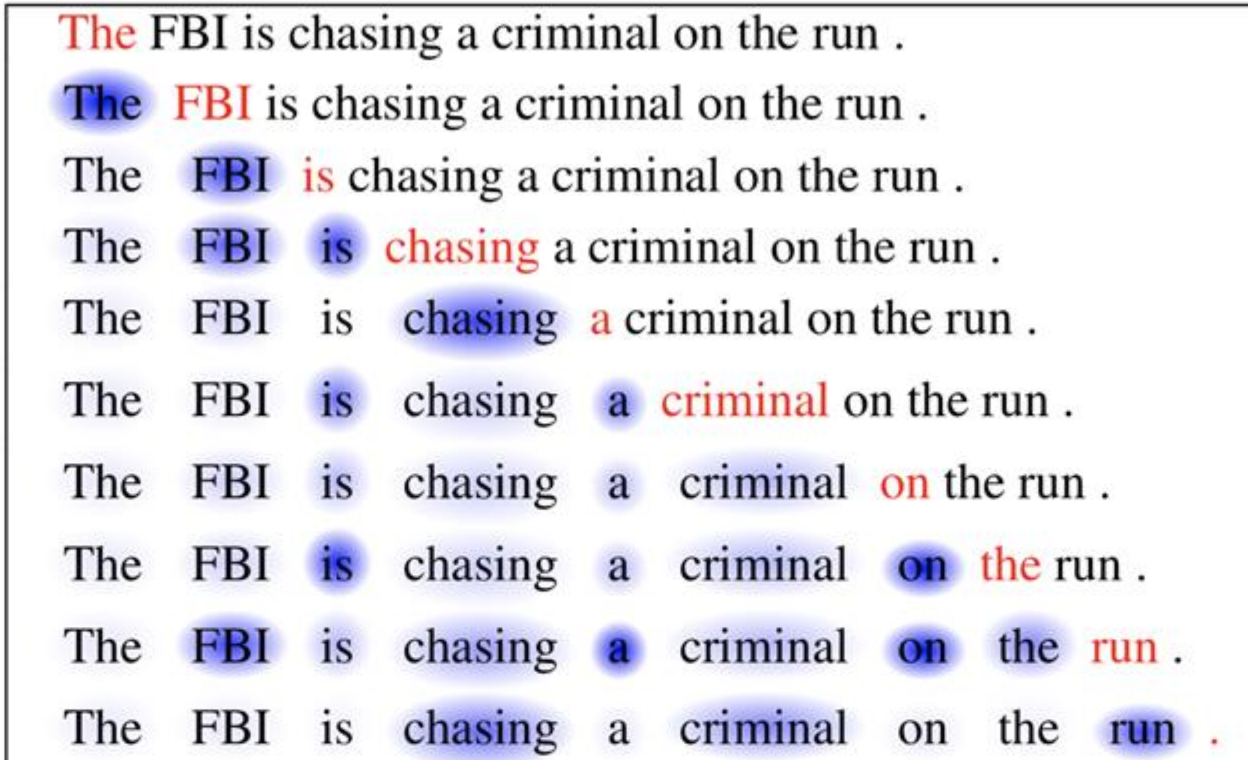


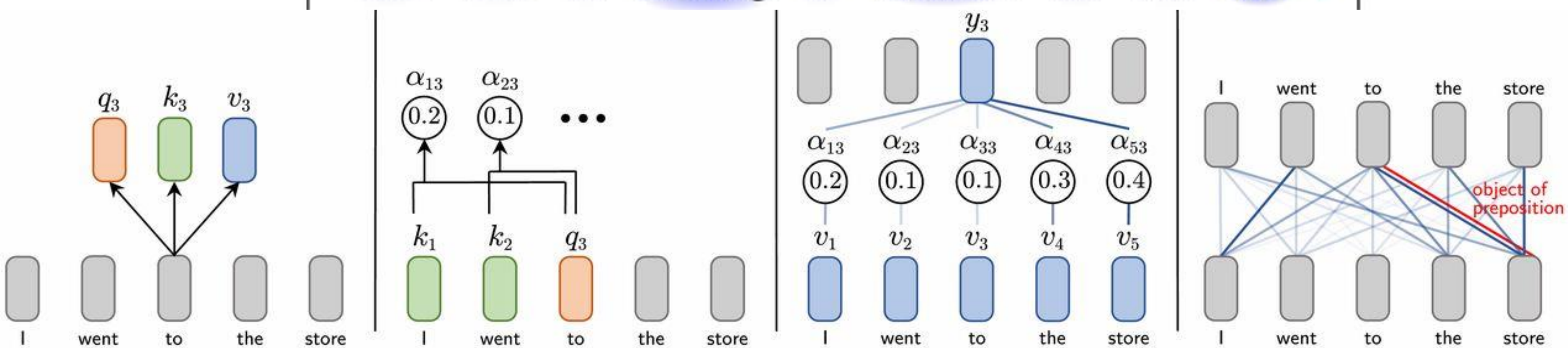
Each encoder / decoder layer has a self-attention layer and a feed forward.



Based: Dr. Yangqiu Song’s slides

Self-attention creates attention layers mapping from a sequence to itself.





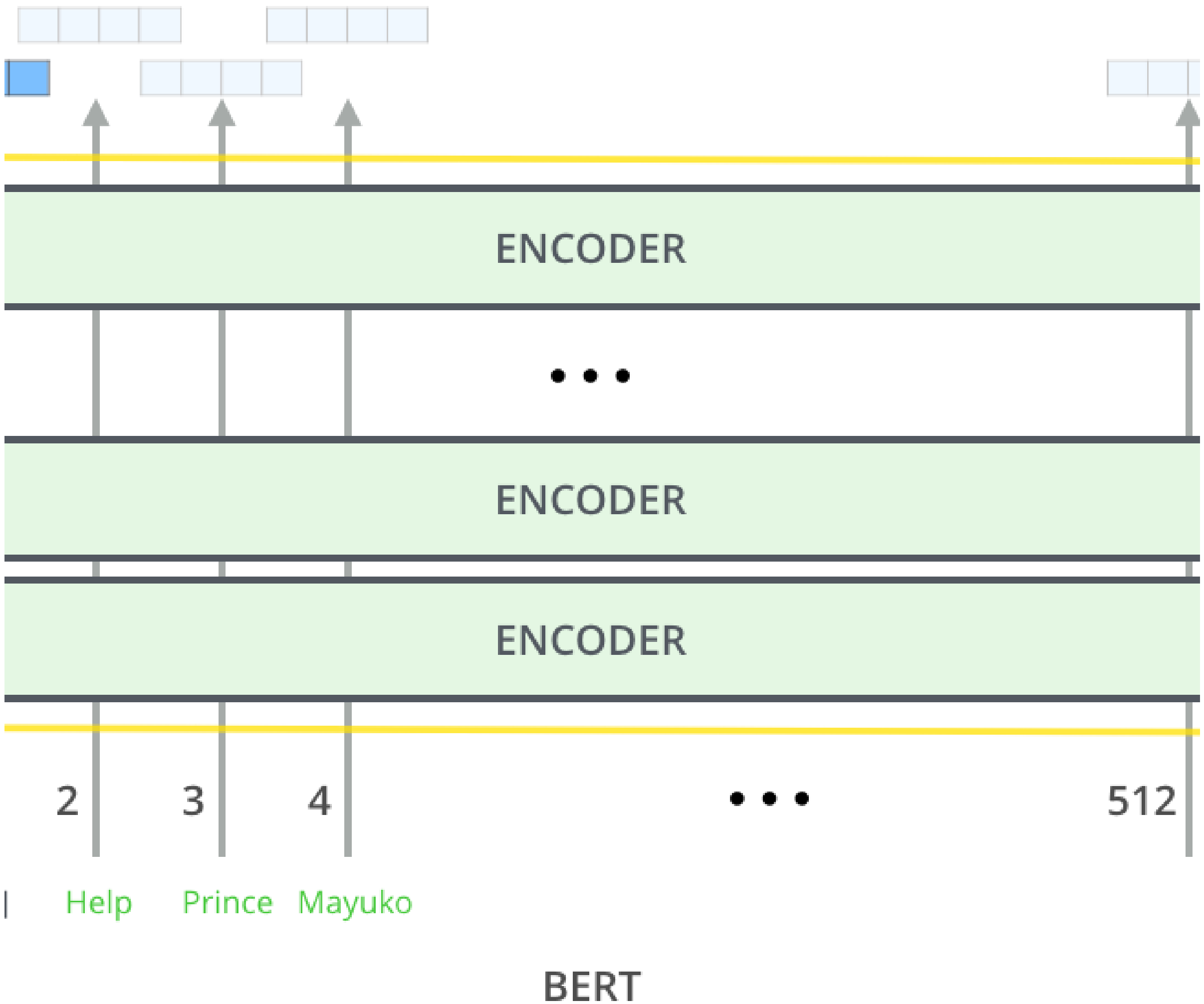
Each input vector is linearly transformed into query, key, and value vectors

Attention weights are normalized inner products of query and key vectors

Outputs are weighted sums of value vectors

After training, the attention weights can be compared with linguistic annotations

BERT: **B**idirectional **E**ncoder **R**epresentations from **Transformers**.



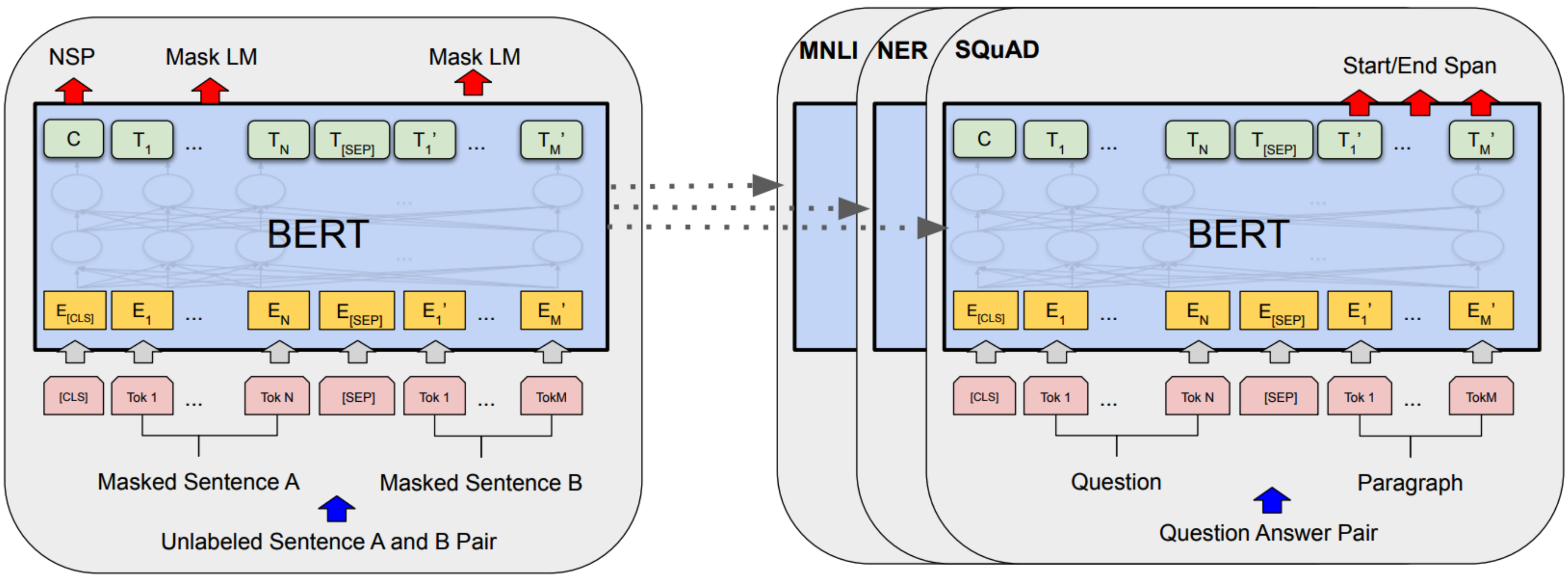
BERT's architecture is just a transformer's encoder stack.

Based: Dr. Yangqiu Song's slides

BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding (NAACL 2019, Devlin et al.)

- Denoising Auto Encoder
- [MASK]: a unique token introduced in the training process to mask some tokens
- Predict masked tokens based on their context information,
- Pre-train and fine-tune

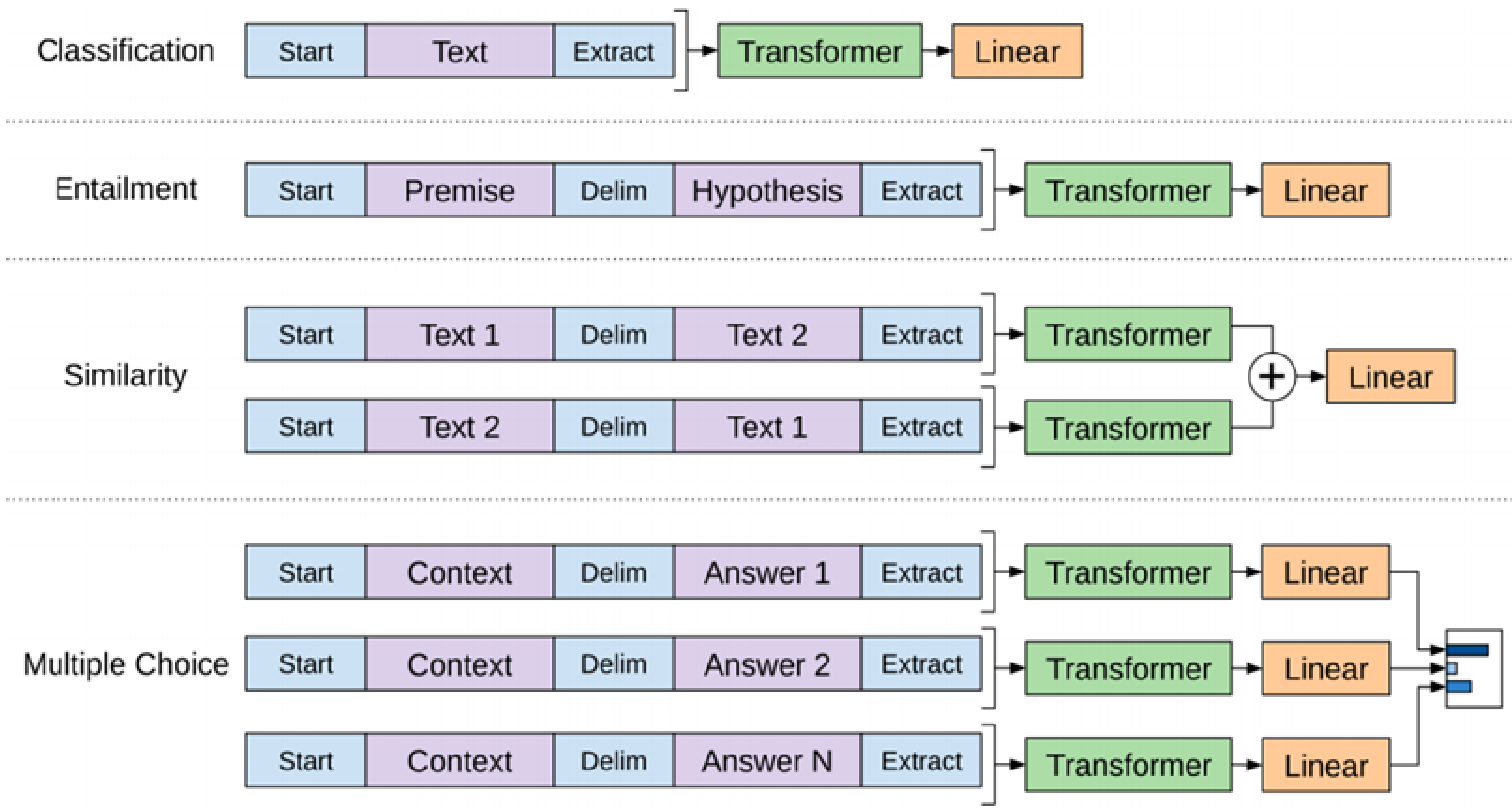
- Intuition: representation should be robust to the introduction of noise
  - Masked Language Model (MLM)



Pre-training Fine-Tuning

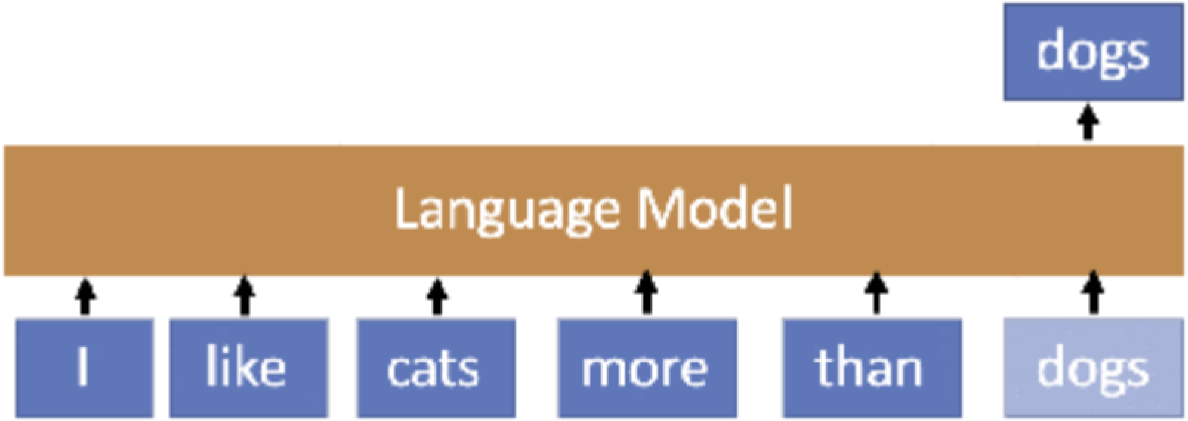
Open AI's GPT-2: 1.5 billion parameters! Trained on 8M pages from reddit

Can use pretrained GPT models for any task. Different tasks use the OpenAI transformer in different ways.



GPT: generative pre-training,

GPT 's architecture is just a transformer's decoder stack.



The prediction scheme for a traditional language model. Shaded words are provided as input to the model while unshaded words are masked out.

Autoregressive Models

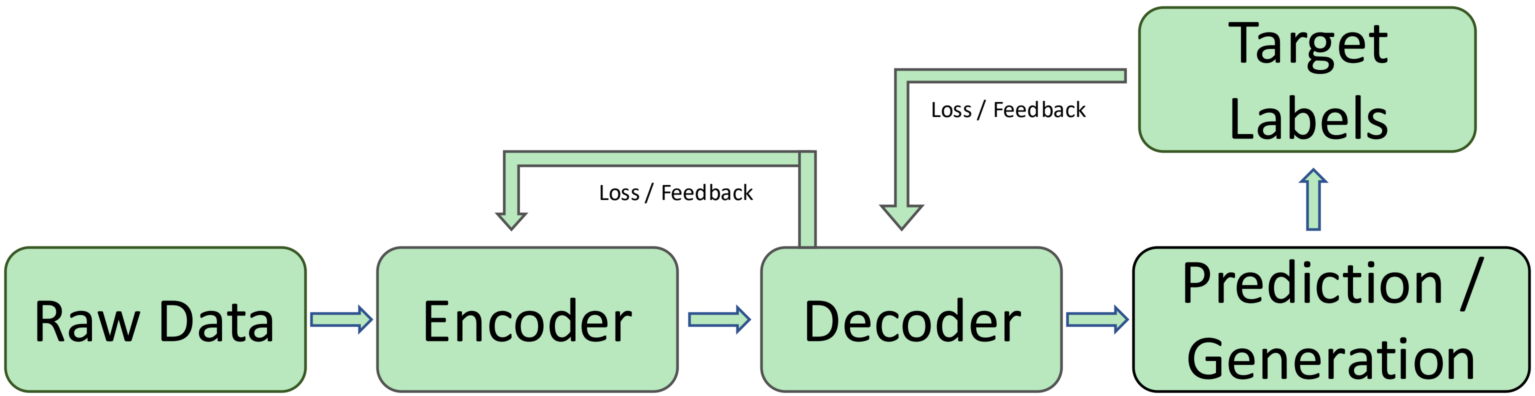
GPT 's pretraining uses next token prediction loss

$$P(x;\theta) = \prod_{n=1}^{N} P(x_n | x_{<n}; \theta)$$

- Each factor can be parametrized by θ, which can be shared.
- The variables can be arbitrarily ordered and grouped, as long as the ordering and grouping is consistent.

Adapt from From NIPS 2017 DL Trend Tutorial

Summary:

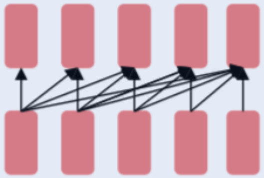


Text
Images
Audio
Discrete/continuous values
Structured/unstructured
Clean/noisy labels

Low dimensional embedding
Category
Generated text, image, audio

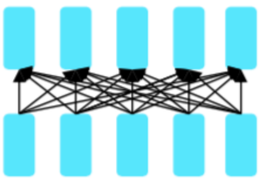
GPT: **G**enerative **P**retraining **M**odels for Language
CLIP: **C**ontrastive **L**anguage-**I**mage **P**retraining for Vision
BERT: **B**idirectional **Encoder R**epresentations from **Transformers**.

Background: Pretraining for three types of architectures



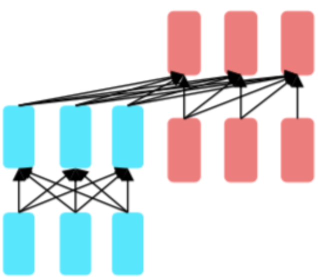
Decoders

- Nice to generate from; can't condition on future words
- **Examples:** GPT-2, GPT-3, LaMDA



Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- **Examples:** BERT and its many variants, e.g. RoBERTa



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?
- **Examples:** T5, Meena

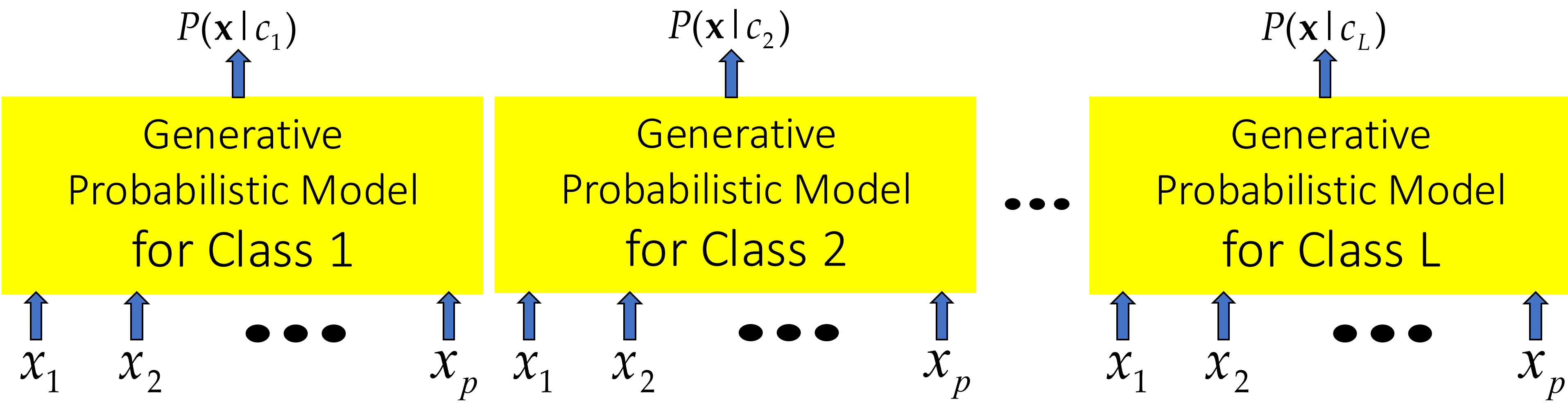
S3: Lecture 16: Generative Bayes Classifiers

- ✓ Bayes Classifier (BC)
  - Generative Bayes Classifier
- ✓ Naïve Bayes Classifier
- Extra ✓ **Gaussian Bayes Classifiers**
  - Gaussian distribution
  - Naïve Gaussian BC
  - Not-naïve Gaussian BC → LDA, QDA

(2) Generative BC

$$\arg\max_{c \in C} P(c / \mathbf{X}), \quad C = \{c_1, \cdots, c_L\}$$

$$P(\mathbf{X}|C), \quad C = c_1, \cdots, c_L, \ \mathbf{X} = (X_1, \cdots, X_p)$$



$$\mathbf{x} = (x_1, x_2, \cdots, x_p)$$

Adapt from Prof. Ke Chen NB slides

Review : Bayes' Rule – for Generative Bayes Classifiers

$$\arg\max_{c \in C} P(c / \mathbf{X}), \quad C = \{c_1, \cdots, c_L\}$$

$$P(C \mid X) = \frac{P(X \mid C) P(C)}{P(X)}$$

Prior

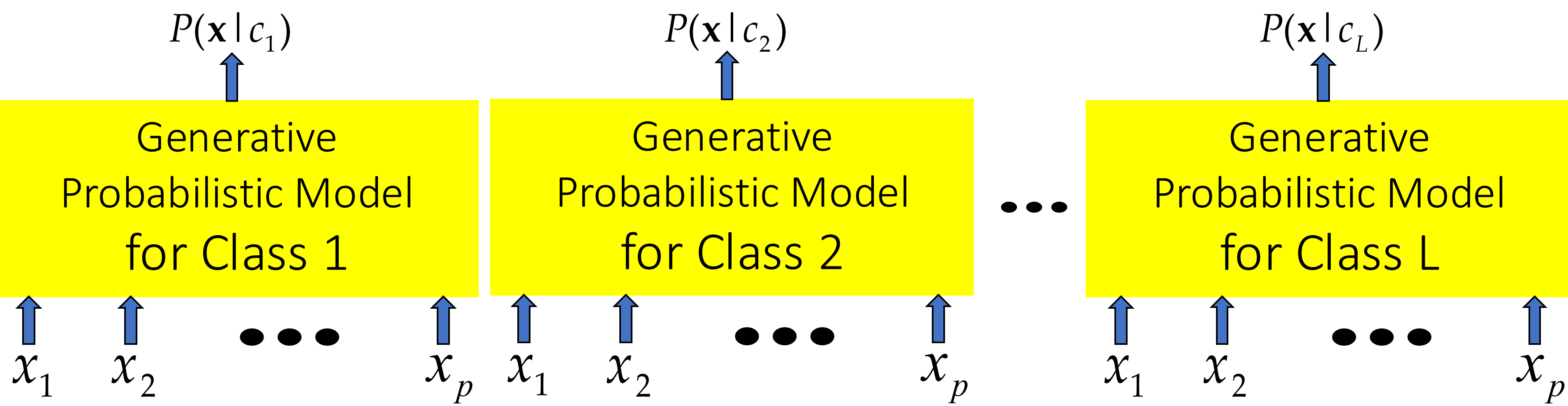
$P(C_1|x)$, $P(C_2|x)$, …, $P(C_L|x)$

$P(C_1)$, $P(C_2)$, …, $P(C_L)$

$$P(C_i \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C_i) P(C_i)}{P(\mathbf{X})}$$

Establishing a probabilistic model for classification through generative probabilistic models

$$\underset{C_i}{\operatorname{argmax}} P(C_i \mid X) = \underset{C_i}{\operatorname{argmax}} P(X, C_i) = \underset{C_i}{\operatorname{argmax}} P(X \mid C_i) P(C_i)$$



$$\mathbf{x} = (x_1, x_2, \cdots, x_p)$$

Adapt from Prof. Ke Chen NB slides

An Example

X_1	X_2	X_3	C

- Example: Play Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

X_1 X_2 X_3 X_4 C

Generative Bayes Classifier:

- Learning Phase ➔ $$\underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_p \mid c_j) P(c_j)$$

$P(C_1), P(C_2), \ldots, P(C_L)$

P(Play=Yes) = 9/14 P(Play=No) = 5/14

$P(X_1, X_2, \ldots, X_p|C_1), P(X_1, X_2, \ldots, X_p|C_2)$

Outlook (3 values)	Temperature (3 values)	Humidity (2 values)	Wind (2 values)	Play=Yes	Play=No
sunny	hot	high	weak	0/9	1/5
sunny	hot	high	strong	.../9	.../5
sunny	hot	normal	weak	.../9	.../5
sunny	hot	normal	strong	.../9	.../5
....	....	....	....	....	....
....	....	....	....	....	....
....	....	....	....	....	....
....	....	....	....	....	....

3*3*2*2 [conjunctions of attributes] * 2 [two classes]=72 parameters

Generative Bayes Classifier:

- Testing Phase
  - Given an unknown instance $\mathbf{X}'_{ts} = (a'_1, \cdots, a'_p)$
  - Look up tables to assign the label c* to X_{ts} if

Last Page: the learned model

$$\hat{P}(a'_1, \cdots, a'_p \mid c^*)\hat{P}(c^*) > \hat{P}(a'_1, \cdots, a'_p \mid c)\hat{P}(c),$$

$$c \neq c^*, \; c = c_1, \cdots, c_L$$

  - Given a new instance,

    x'=(Outlook=Sunny, Temperature=Cool, Humidity=High, Wind=Strong)

Naïve Bayes Classifier

- Naïve Bayes classification
  - Assumption that all input attributes are conditionally independent!

[assumption]

$$P(X_1, X_2, \cdots, X_p | C) = P(X_1|C)P(X_2|C) \cdots P(X_p|C)$$

Counting

- Learning Phase

$Estimate\ P(X_j = x_{jk} | C = c_i)\ with\ examples\ in\ training;$

$P(X_2|C_1), P(X_2|C_2)$

Outlook	Play=Yes	Play=No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=Yes	Play=No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

$P(X_4|C_1), P(X_4|C_2)$

Humidity	Play=Yes	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=Yes	Play=No
Strong	3/9	3/5
Weak	6/9	2/5

3+3+2+2 [naïve assumption] * 2 [two classes]= 20 parameters

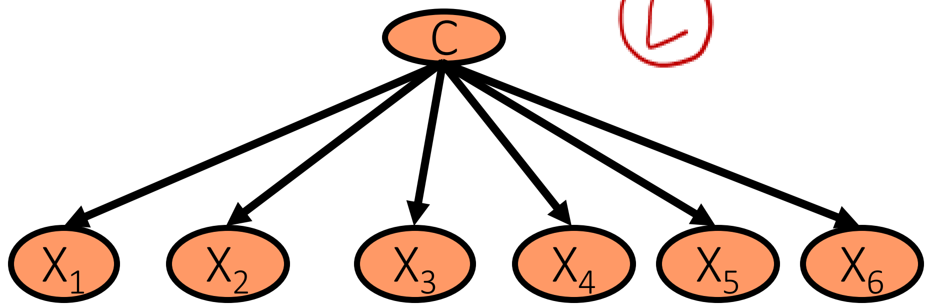
P(Play=Yes) = 9/14   P(Play=No) = 5/14

$P(C_1), P(C_2), \ldots, P(C_L)$

Learning (training) the NBC Model

$P(c_1), \dots, P(c_L)$

L



- maximum likelihood estimates:
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

$$[\hat{P}(a'_1|c^*)\cdots\hat{P}(a'_p|c^*)]\hat{P}(c^*) > [\hat{P}(a'_1|c)\cdots\hat{P}(a'_p|c)]\hat{P}(c)$$

- Test Phase
  - Given a new instance,

    x'=(Outlook=Sunny, Temperature=Cool, Humidity=High, Wind=Strong)
  - Look up in conditional-prob tables

P(Outlook=Sunny|Play=Yes) = 2/9	P(Outlook=Sunny|Play=No) = 3/5
P(Temperature=Cool|Play=Yes) = 3/9	P(Temperature=Cool|Play==No) = 1/5
P(Huminity=High|Play=Yes) = 3/9	P(Huminity=High|Play=No) = 4/5
P(Wind=Strong|Play=Yes) = 3/9	P(Wind=Strong|Play=No) = 3/5
P(Play=Yes) = 9/14	P(Play=No) = 5/14

  - MAP rule

    P(Yes|x'): [P(Sunny|Yes)P(Cool|Yes)P(High|Yes)P(Strong|Yes)]P(Play=Yes) = 0.0053

    P(No|x'): [P(Sunny|No) P(Cool|No)P(High|No)P(Strong|No)]P(Play=No) = 0.0206

Given the fact P(Yes|x') < P(No|x'), we label x' to be "No".

Summary:
Generative Bayes Classifier

Task: Classify a new instance *X* based on a tuple of attribute values $X = \langle X_1, X_2, \ldots, X_p \rangle$ into one of the classes

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_p)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, \ldots, x_p \mid c_j) P(c_j)}{P(x_1, x_2, \ldots, x_p)}$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_p \mid c_j) P(c_j)$$

$j = 1, 2, .., L$

MAP = Maximum A Posteriori

Adapt From Carols' prob tutorial

WHY ? Naïve Bayes Assumption

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples.

Not Naïve

- $P(x_1,x_2,\ldots,x_p/c_j)$
  - $O(/X_1/.\ /X_2/.\ /X_3/\ldots.\ /X_p/./C/)$ parameters
  - Could only be estimated if a very, very large number of training examples was available.

Naïve

- $P(x_k/c_j)$
  - $O([/X_1/+\ /X_2/+\ /X_3/\ldots.+\ /X_p/]./C/)$ parameters
  - Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i|c_j)$.

Smoothing to Avoid Overfitting

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k_i}$$

# of values of X_i

- Somewhat more subtle version

overall fraction in data where $X_i = x_{i,k}$

$$\hat{P}(x_{i,k} \mid c_j) = \frac{N(X_i = x_{i,k}, C = c_j) + m p_{i,k}}{N(C = c_j) + m}$$

$k \in \{1, 2, \ldots, k_i\}$

extent of "smoothing"

11/06

Today Roadmap

- 1. TA Kefang going over the HW3 solutions
- 2. Prof. Qi to quickly review some course content
  - 2a. the LLM-advanced-survey slide deck (URL)
  - 2b. the NBC for text + SVM basics (move to 11/11, due to project meetings)
- 3. Quiz 10

11/11

Today Roadmap

- 0. A few announcements:
  - HW4 / HW5 / Project due time
  - Survey 3 / Nov.25th class zoom or in-person?
- 1. Prof. Qi to quickly review some course content
  - A. the NBC for text + SVM basics
  - B. the 10 advanced topics on deep learning (TBD)
- 2. Quiz 11

A few announcements

- HW4 / HW5 / Project due time
  - HW4 due moves to: Nov. 15th midnight
  - HW5 due moves to: Dec. 1st midnight
  - Project deliverables:
    - Slide deck and code to be submitted by Dec. 16th midnight
    - Final presentation dates moves to zoom on Dec. 11th / 12th / 15th / 16th (every team has a 25mins session, so a total of 12 hours online sessions --- these sessions are open to everybody in the class!!!)

- Added two more office hours (4 from TA, 1 from me)
- Final Exam: Dec. 9th in class exam
  - We will host course review sessions on Dec. 2nd and 4th

- Survey 3 needs your inputs on:
  - Nov.25th class zoom or in-person?
  - Overall satisfaction of the course experience so far..

Review: Naïve Bayes Classifier

$$\underset{C}{\operatorname{argmax}} P(C \mid X) = \underset{C}{\operatorname{argmax}} P(X, C) = \underset{C}{\operatorname{argmax}} P(X \mid C) P(C)$$

Naïve Bayes Classifier

$$P(X_1, X_2, \cdots, X_p | C) = P(X_1|C) P(X_2|C) \cdots P(X_p|C)$$

$$c^* = argmax \; P(C = c_i | \mathbf{X} = \mathbf{x}) \propto P(\mathbf{X} = \mathbf{x} | C = c_i) P(C = c_i)$$

for $i = 1, 2, \cdots, L$

Assuming all input attributes are conditionally independent given a specific class label!

$$\underset{k}{\operatorname{argmax}} P(C\_k \mid X) = \underset{k}{\operatorname{argmax}} P(X, C) = \underset{k}{\operatorname{argmax}} P(X \mid C) P(C)$$

Generative Bayes Classifiers

Task → Representation → Score Function → Search/Optimization → Models, Parameters

classification → Prob. models p(X|C) → EPE with 0-1 loss → MAP Rule → Many options → Prob. Models' Parameter

$$P(X_1, \ldots, X_p \mid C)$$

(EXTRA)

Gaussian Naive

$$\hat{P}(X_j \mid C = c_k) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp\left(-\frac{(X_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right)$$

Bernoulli Naïve

$$p(W_i = true \mid c_k) = p_{i,k}$$

Multinomial

$$P(W_1 = n_1, \ldots, W_v = n_v \mid c_k) = \frac{N!}{n_{1k}! n_{2k}! .. n_{vk}!} \theta_{1k}^{n_{1k}} \theta_{2k}^{n_{2k}} .. \theta_{vk}^{n_{vk}}$$

L17 : Naïve Bayes Classifier for Text

- ✓ Dictionary based Vector space representation of text article
- ✓ Multivariate Bernoulli vs. Multinomial
- ✓ Multivariate Bernoulli naïve Bayes classifier
  - Testing
  - Training With Maximum Likelihood Estimation for estimating parameters
- ✓ Multinomial naïve Bayes classifier
  - Testing
  - Training With Maximum Likelihood Estimation for estimating parameters
  - Multinomial naïve Bayes classifier as Conditional Stochastic Language Models (Extra)

The bag of words representation

$f($ I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet. $) = c$

The bag of words representation

f(

great	2
love	2
recommend	1
laugh	1
happy	1
...	...

)=c

Common refinements: remove stopwords, stemming, collapsing multiple occurrences of words into one….

Unknown Words

- How to handle words in the test corpus that did not occur in the training data, i.e. out of vocabulary (OOV) words?
- Train a model that includes an explicit symbol for an unknown word (<UNK>).
  - Choose a vocabulary in advance and replace other (i.e. not in vocabulary) words in the corpus with <UNK>.
  - Very often, <UNK> also used to replace rare words

Naïve Probabilistic Models of text documents

$$\Pr(D \mid C = c) =$$

Two Previous NBC models

$$\Pr(W_1 = true, W_2 = false \ldots, W_k = true \mid C = c)$$

Multivariate Bernoulli Distribution

$$\Pr(W_1 = n_1, W_2 = n_2, \ldots, W_k = n_k \mid C = c)$$

Multinomial Distribution

Model 1: Multivariate Bernoulli

- Model 1: Multivariate Bernoulli
  - For each word in a dictionary, feature
  - X_w = true in document d if w appears in d
  - Naive Bayes assumption:
    - Given the document's topic class label, appearance of one word in the document tells us nothing about chances that another word appears

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun… It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

word	Boolean
great	Yes
love	Yes
recommend	Yes
laugh	Yes
happy	Yes
hate	No

X_w

Model 2: Multinomial Naïve Bayes - 'Bag of words' representation of text

word	frequency
great	2
love	2
recommend	1
laugh	1
happy	1
...	.

$$\Pr(W_1 = n_1, \ldots, W_k = n_k \mid C = c)$$

Can be represented as a multinomial distribution.

Words = like colored balls, there are K possible type of them (i.e. from a dictionary of K words)

A Document = contains N words, each word occurs n_i times (like a bag of N colored balls)

Training: Parameter estimation

Multinomial model:

$$\hat{P}(X_i = w_i | c_j) = \quad \theta_{w_i, c_j}$$

fraction of times in which each dictionary word w appears across all documents of class c_j

- Can create a mega-document for class j by concatenating all documents on this class,
- Use frequency of w in mega-document

Adapt From Manning' textCat tutorial

Model 2: Multinomial Naïve Bayes - 'Bag of words' – TESTING Stage

word	frequency
great	2
love	2
recommend	1
laugh	1
happy	1
...	.

$$\arg\max_c P(W_1 = n_1, \ldots, W_k = n_k, c)$$

$$= \arg\max_c \{p(c) * \theta_{1,c}^{\ n_1} \theta_{2,c}^{\ n_2} .. \theta_{k,c}^{\ n_k}\}$$

L20: Basic Support Vector Machine

Data → Task → Representation → Score Function → Search/Optimization → Models, Parameters

Tabular → Classification → Kernel Trick Func K(xi, xj) → Margin + Hinge Loss → QP with Dual form → Dual Weights

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

$$w = \sum_i \alpha_i x_i y_i$$

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + C \sum_{i=1}^{n} \varepsilon_i$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 1 - \varepsilon_i$$

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

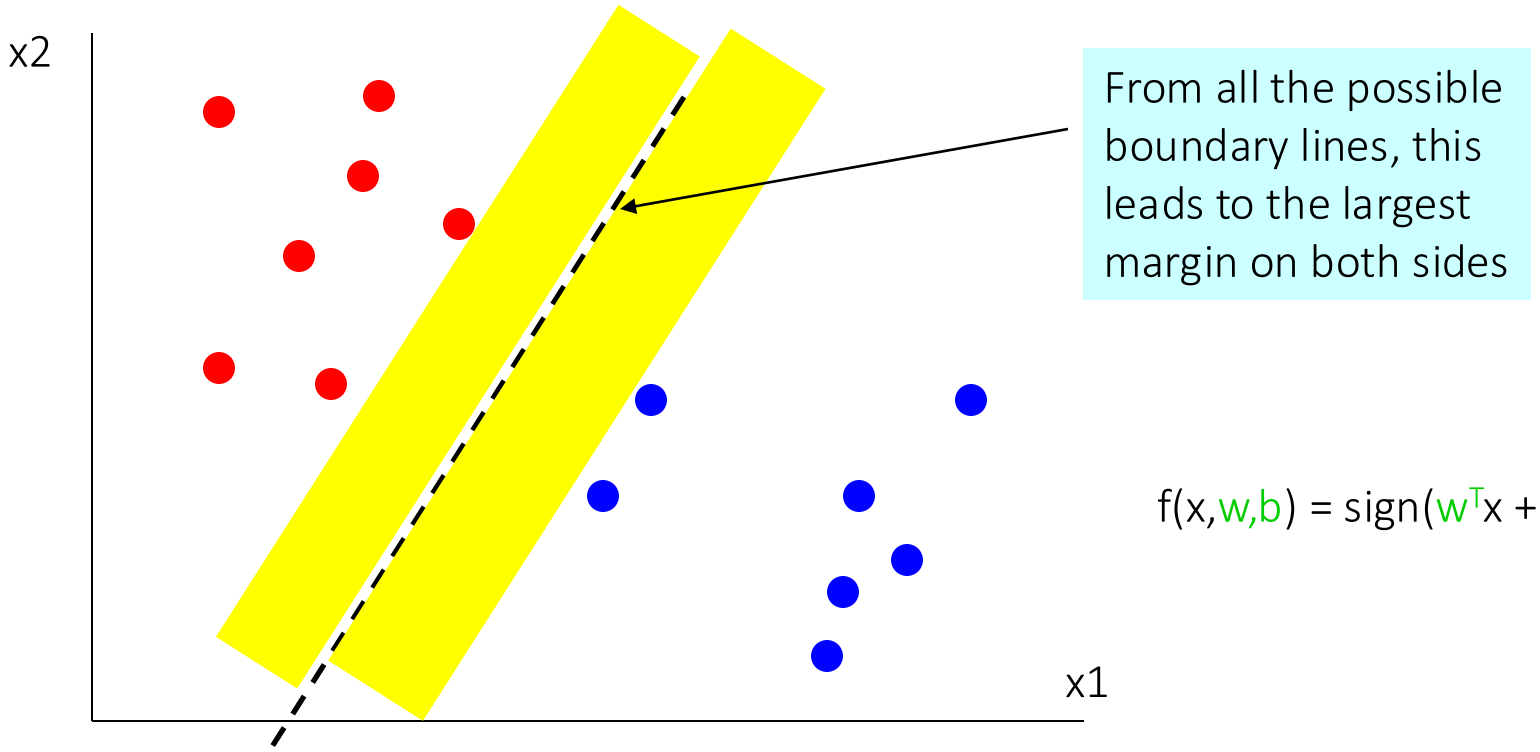
$$\sum_i \alpha_i y_i = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

MNIST

Classifier	Test Error Rate (%)	References
Linear classifier (1-layer neural net)	12.0	LeCun et al. (1998)
K-nearest-neighbors, Euclidean (L2)	5.0	LeCun et al. (1998)
2-Layer neural net, 300 hidden units, mean square error	4.7	LeCun et al. (1998)
Support vector machine, Gaussian kernel	1.4	MNIST Website
Convolutional net, LeNet-5 (no distortions)	**0.95**	LeCun et al. (1998)
Methods using distortions		
Virtual support vector machine, deg-9 polynomial, (2-pixel jittered and deskewing)	0.56	DeCoste and Scholkopf (2002)
Convolutional neural net (elastic distortions)	0.4	Simard, Steinkraus, and Platt (2003)
6-Layer feedforward neural net (on GPU) (elastic distortions)	**0.35**	Ciresan, Meier, Gambardella, and Schmidhuber (2010)
Large/deep convolutional neural net (elastic distortions)	**0.35**	Ciresan, Meier, Masci, Maria Gambardella, and Schmidhuber (2011)
Committee of 35 convolutional networks (elastic distortions)	0.23	Ciresan, Meier, and Schmidhuber (2012)

Max margin classifiers

- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest margin from both sets of points

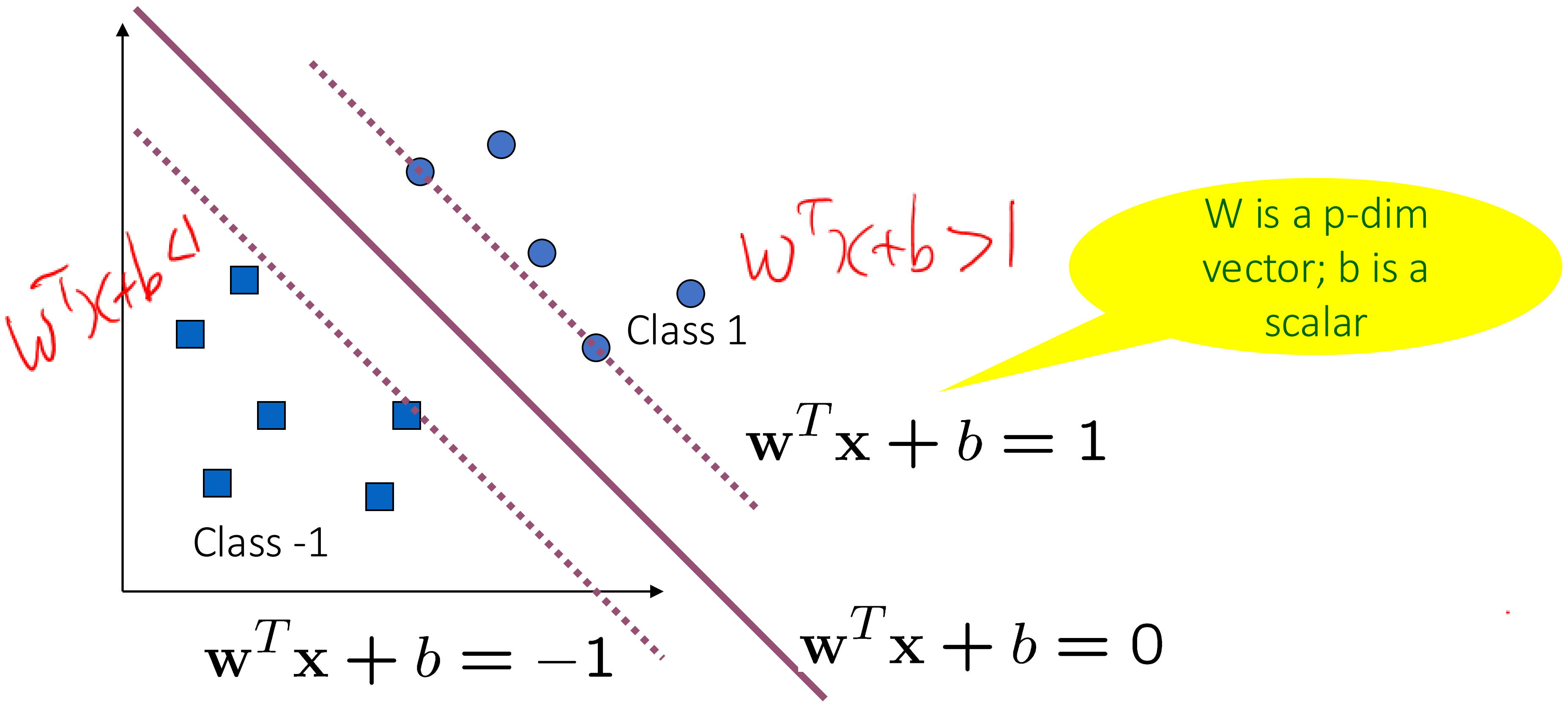


$f(x,w,b) = \text{sign}(w^T x + b)$

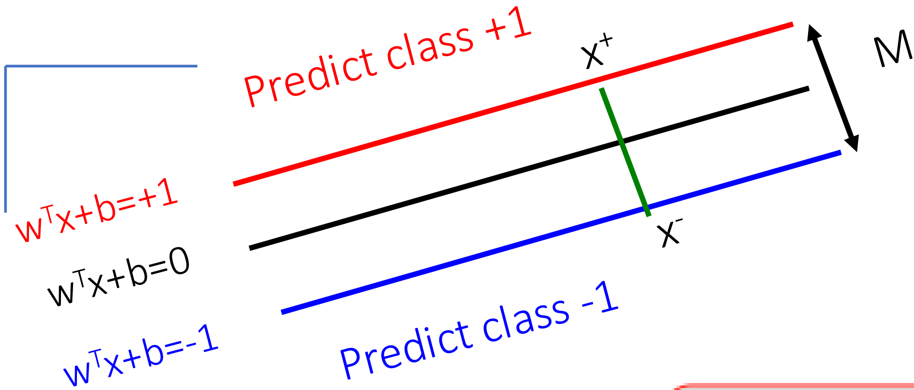
Credit: Prof. Moore

Max-margin & Decision Boundary $\mathbf{w}^T\mathbf{x} + b = 0$

- The decision boundary should be as far away from the data of both classes as possible



Optimization Step
i.e. learning optimal parameter for SVM



$$M = \frac{2}{\sqrt{w^T w}}$$

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes w^Tw)

Min $(w^Tw)/2$

subject to the following constraints:

For all x in class + 1

$w^Tx+b >= 1$

For all x in class - 1

$w^Tx+b <= -1$

A total of n constraints if we have n training samples

Optimization Reformulation

$f(x,w,b) = \mathrm{sign}(w^T x + b)$

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

$y_i \in \{+1, -1\}$

Quadratic Objective

Min $(w^T w)/2$

subject to the following constraints:

For all x in class + 1

$w^T x + b >= 1$

For all x in class - 1

$w^T x + b <= -1$

A total of n constraints if we have n input samples

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : \underbrace{y_i}_{1\times1} \Big(\underbrace{\underbrace{\mathbf{w}^T}_{1\times p} \underbrace{\mathbf{x}_i}_{p\times1}}_{1\times1} + \underbrace{b}_{1\times1} \Big) \geq 1$$

Quadratic programming i.e.,
- Quadratic objective
- Linear constraints

Two optimization problems: For the separable and non separable cases

Min $(w^T w)/2$

For all x in class + 1

$w^T x + b >= 1$

For all x in class - 1

$w^T x + b <= -1$

$$\min_w \frac{w^T w}{2} + C \sum_{i=1}^{n} \varepsilon_i$$

For all x_i in class + 1

$w^T x_i + b >= 1 - \varepsilon_i$

For all x_i in class - 1

$w^T x_i + b <= -1 + \varepsilon_i$

For all i

$$\varepsilon_i \geq 0$$

- Instead of solving these QPs directly we will solve a dual formulation of the SVM optimization problem
- The main reason for switching to this type of representation is that it would allow us to use a neat trick that will make our lives easier (and the run time faster)

L21: Kernel Support Vector Machine

Data
↓
Task
↓
Representation
↓
Score Function
↓
Search/Optimization
↓
Models, Parameters

Tabular / Text / Image/ Structured
↓
Classification / Regression / Ranking
↓
Kernel Trick Func K(xi, xj)

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

↓
Margin + Hinge Loss
↓
QP with Dual form

$$w = \sum_i \alpha_i x_i y_i$$

↓
Dual Weights

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + C \sum_{i=1}^{n} \varepsilon_i$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 1 - \varepsilon_i$$

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

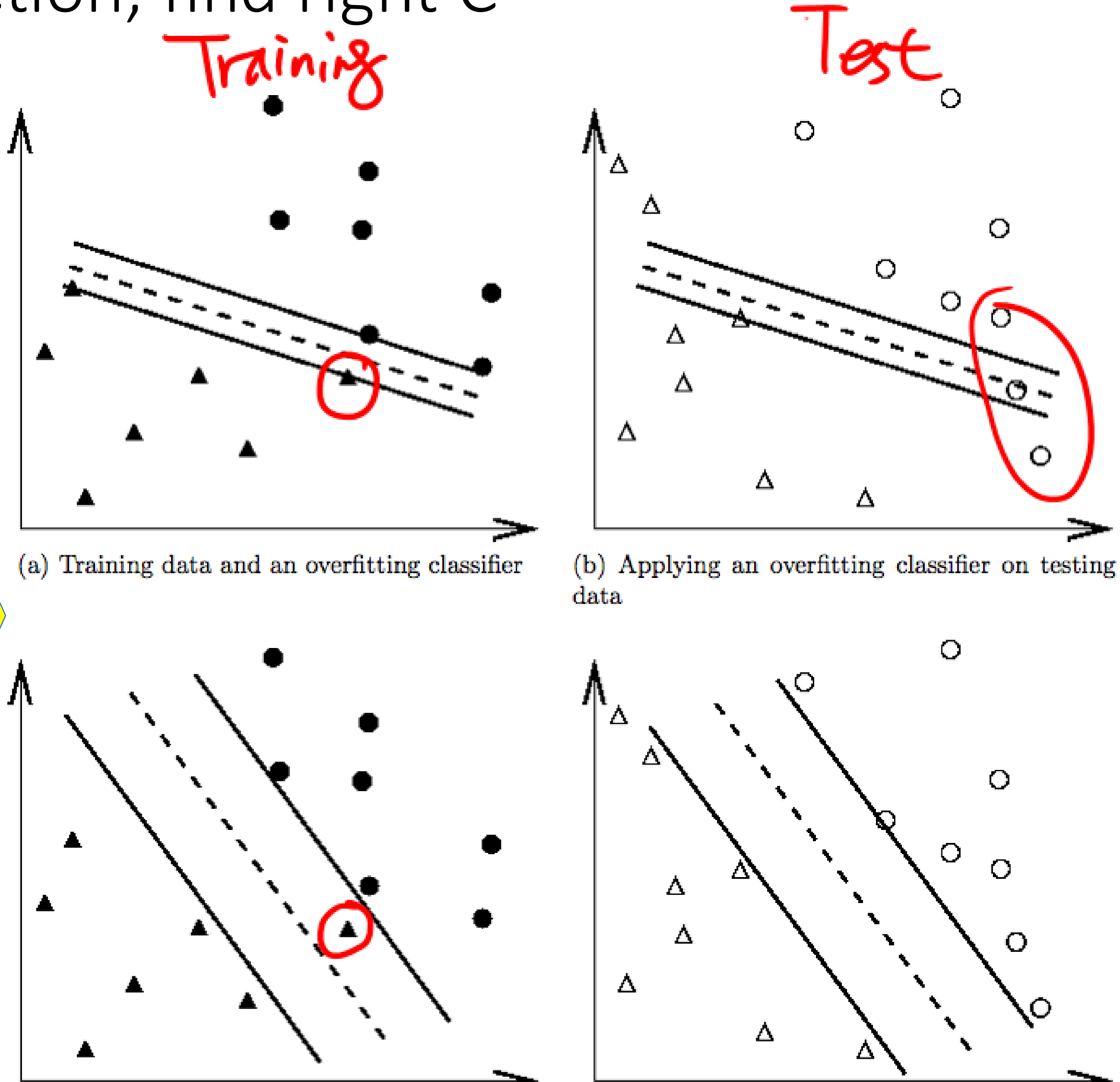
$$\sum_i \alpha_i y_i = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

Model Selection, find right C

large C

Select the right penalty parameter C

small C



(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier

(d) Applying a better classifier on testing data

Model Selection, find right C

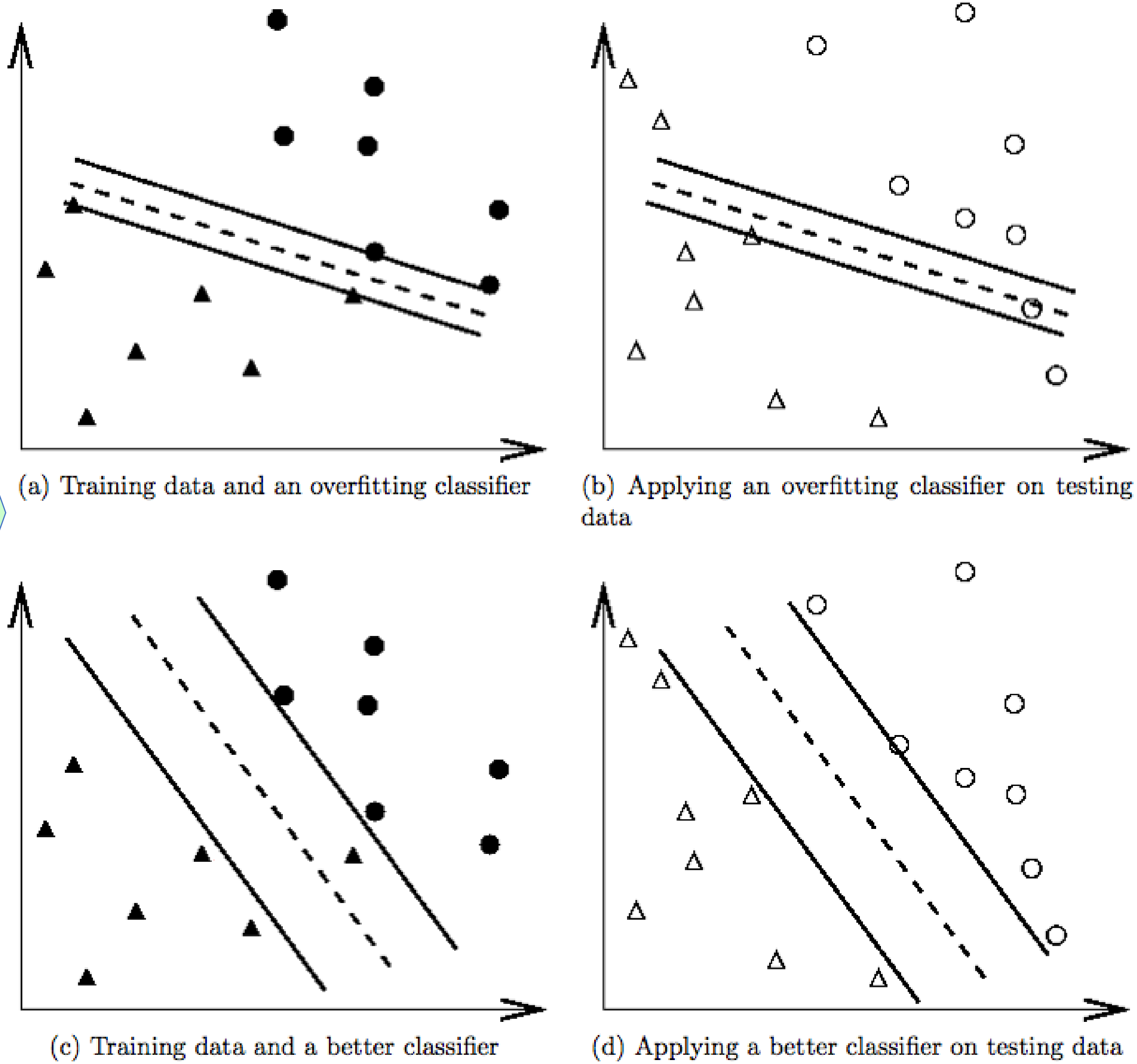
large C

A large value of C means that misclassifications are bad - resulting in smaller margins and less training error (but more expected true error).

may lead

A small C results in more training error, hopefully better true error.

small C



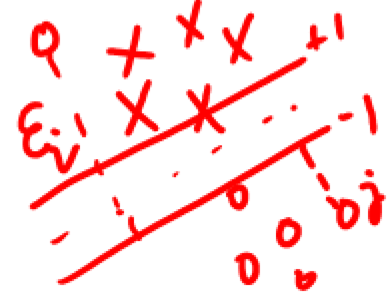
(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier

(d) Applying a better classifier on testing data

$$\sum_{i=1}^{n} \underbrace{\varepsilon_i}$$



Hinge Loss for Soft SVM

$$\sum_{i=1}^{n} \max(0, 1 - y_i f(\mathbf{x}_i))$$

$$\min_{w} \frac{w^T w}{2} + C \sum_{i=1}^{n} \varepsilon_i$$

For all x_i in class + 1

$w^T x_i + b \geq 1 - \varepsilon_i$

For all x_i in class - 1

$w^T x_i + b \leq -1 + \varepsilon_i$

For all i

$\varepsilon_i \geq 0$

soft

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \frac{w^T w}{2} + C \sum_{i=1}^{n} \max(0, 1 - \underbrace{y_i\left(\mathbf{w}^T \mathbf{x}_i + b\right)}_{\geq 1})$$

subject to:

$$y_i\left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 - \varepsilon_i$$

$$\varepsilon_i \geq 0$$

vs. Hard SVM

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 / 2$$

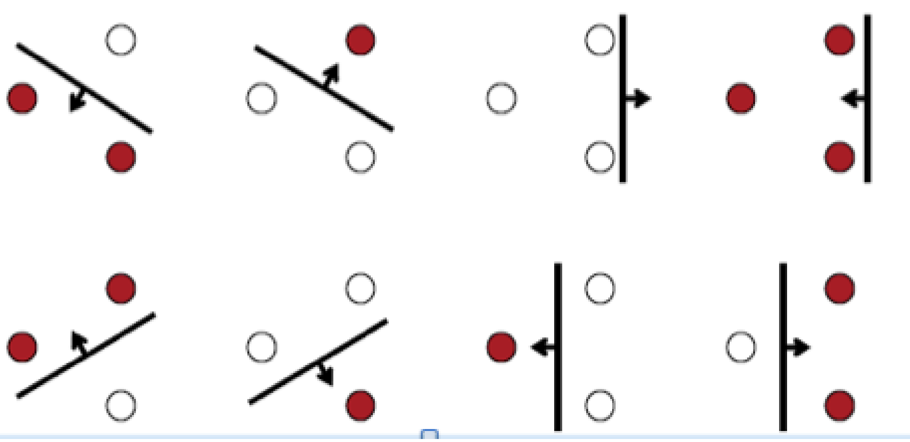
subject to $\forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1$

A little bit theory:
Vapnik-Chervonenkis (VC) dimension

$X \longrightarrow \phi(x)$
$\geq N-1$

If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

- VC dimension of the set of oriented lines in R^2 is 3
  - It can be shown that the VC dimension of the family of oriented separating hyperplanes in R^N is at least N+1

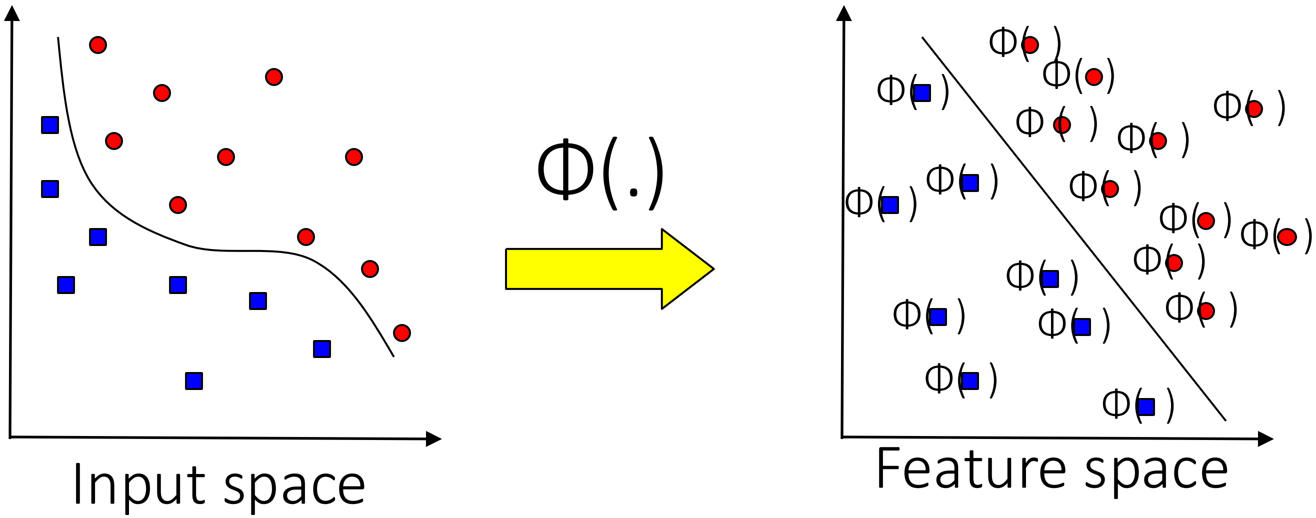


If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;

N data points are in general separable in a space of N-1 dimensions or more!!!

$$X \longrightarrow \Phi(X)$$

linearly separated into two classes $\{+1, -1\}$



SVM solves these two issues simultaneously

- "Kernel tricks" for efficient computation
- Dual formulation only assigns parameters to samples, not to features

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \text{ is called the kernel function.}$$

- Linear kernel (we've seen it)

$$K(\mathbf{x}, z) = \mathbf{x}^T z \qquad \begin{cases} x \in R^p \\ z \in R^p \end{cases}$$

- Polynomial kernel (we will see an example)

$$K(\mathbf{x}, z) = \left(1 + \mathbf{x}^T z\right)^d = \underbrace{\Phi_p(x)^T \Phi_p(z)}$$

$O(p)$

where d = 2, 3, … To get the feature vectors we concatenate all dth order polynomial terms of the components of x (weighted appropriately)

$p_\Phi \to O(p^d)$

- Radial basis kernel

$$K(\mathbf{x}, z) = \exp\left(-r\left\|\mathbf{x} - z\right\|^2\right) = \underbrace{\Phi_r(x)^T}\,\Phi_r(z)$$

$O(p)$ $\qquad p_\Phi = \infty$

In this case., r is hyperpara. The feature space of the RBF kernel has an infinite number of dimensions

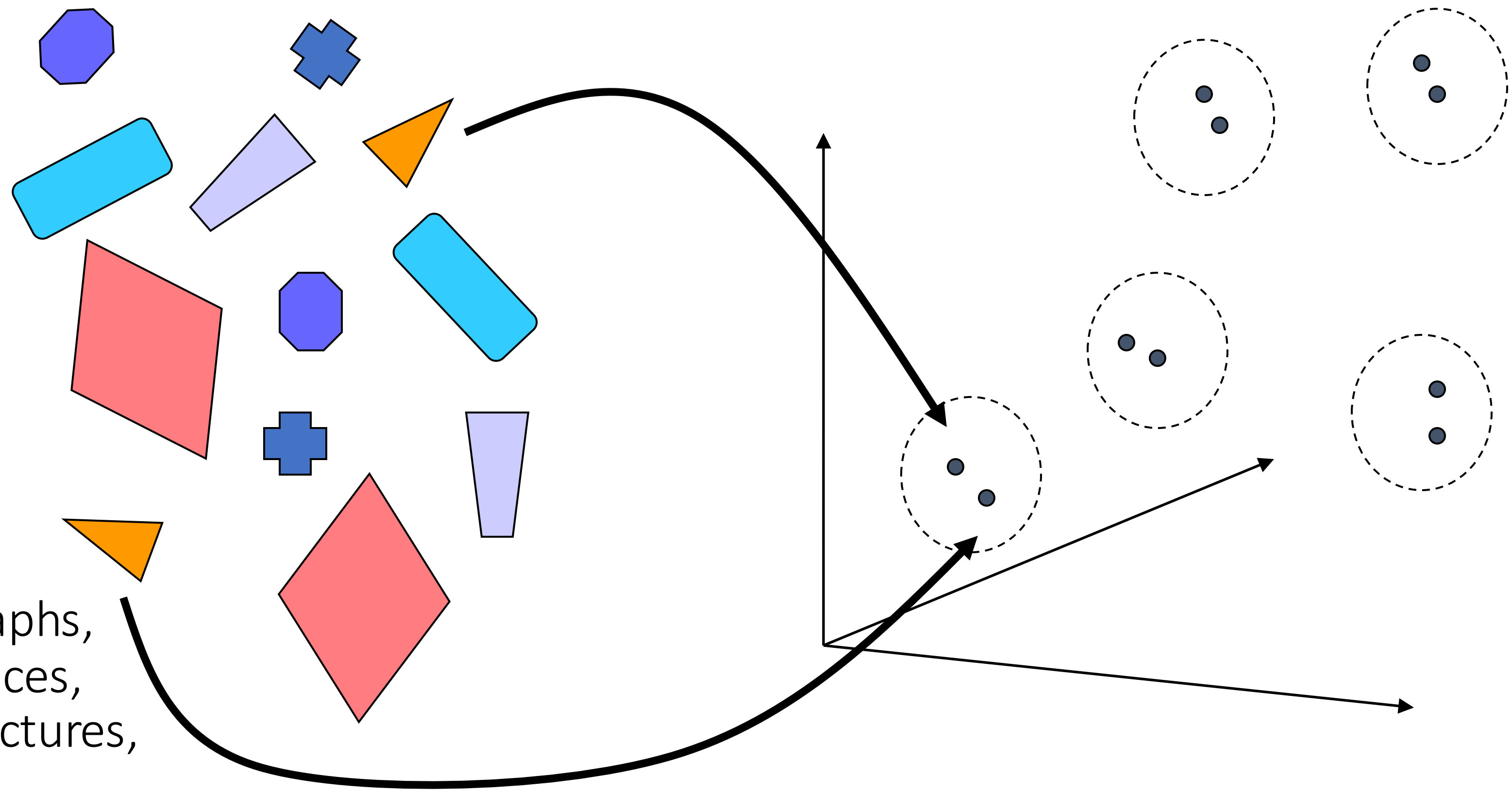
Never represent features explicitly
⍰ Compute dot products with a closed form
Very interesting theory – Reproducing Kernel Hilbert Spaces
⍰ Not covered in detail here

Kernel trick has helped Non-traditional data like strings and trees able to be used as input to SVM, instead of feature vectors

When numerical x and z do not exist, we can calculate

$k(x, z)$

Vector vs. Relational data



e.g. Graphs, Sequences, 3D structures,

Original Space

Feature Space

Summary:
Modification Due to Kernel Trick

- Change all inner products to kernel functions
- For training,

Original Linear

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_{i} \alpha_i \mathrm{y_i} = 0$$

$$C > \alpha_i \geq 0, \forall i \in train$$

With kernel function - nonlinear

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j K(\mathbf{x_i}, \mathbf{x_j})$$

VC: $\phi(x)$ large

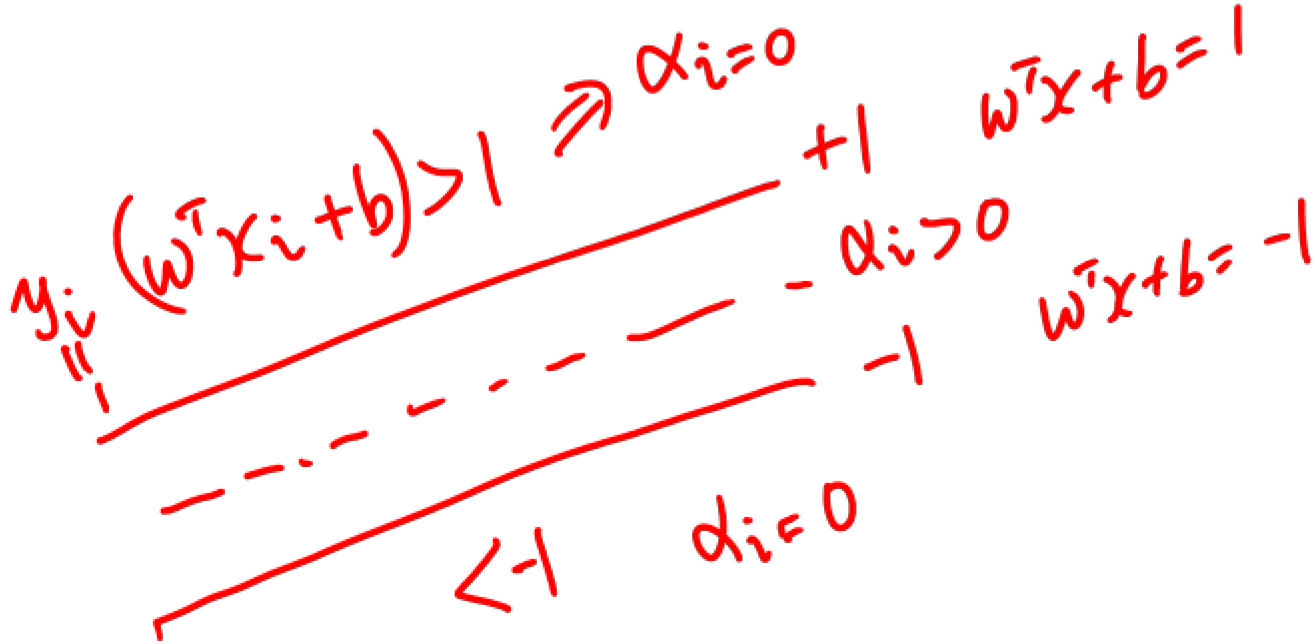
$$\sum_{i} \alpha_i \mathrm{y_i} = 0$$

$$C > \alpha_i \geq 0, \forall i \in train$$

Support vectors: non-zero a_i

- only a few a_i can be nonzero!!

$$\forall i \Rightarrow \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) = 0, \quad i = 1, \dots, n$$



Dual SVM– Testing

Dot product with ("all" ??) training samples

To evaluate a new sample x_{ts} we need to compute:

$$\widehat{y_{ts}} = \mathrm{sign}(w^T x_{ts} + b) = \mathrm{sign}\left(\sum_{i=1..n} \alpha_i y_i \mathbf{x_i}^T \mathbf{x}_{ts} + b\right)$$

$$\Rightarrow \widehat{y_{ts}} = \mathrm{sign}\left(\sum_{i \in SupportVectors} \alpha_i y_i \left(\mathbf{x}_i^T \mathbf{x}_{ts}\right) + b\right)$$

$\Downarrow$

$\alpha_i > 0$

For \alpha$_i$ that are 0,

no influence

Why do SVMs work?

$x \rightarrow \phi(x)$ e.g. RBF

- If we are using huge features spaces (e.g., with kernels), how come we are not overfitting the data?

  ✓ Number of parameters remains the same (and most are set to 0)

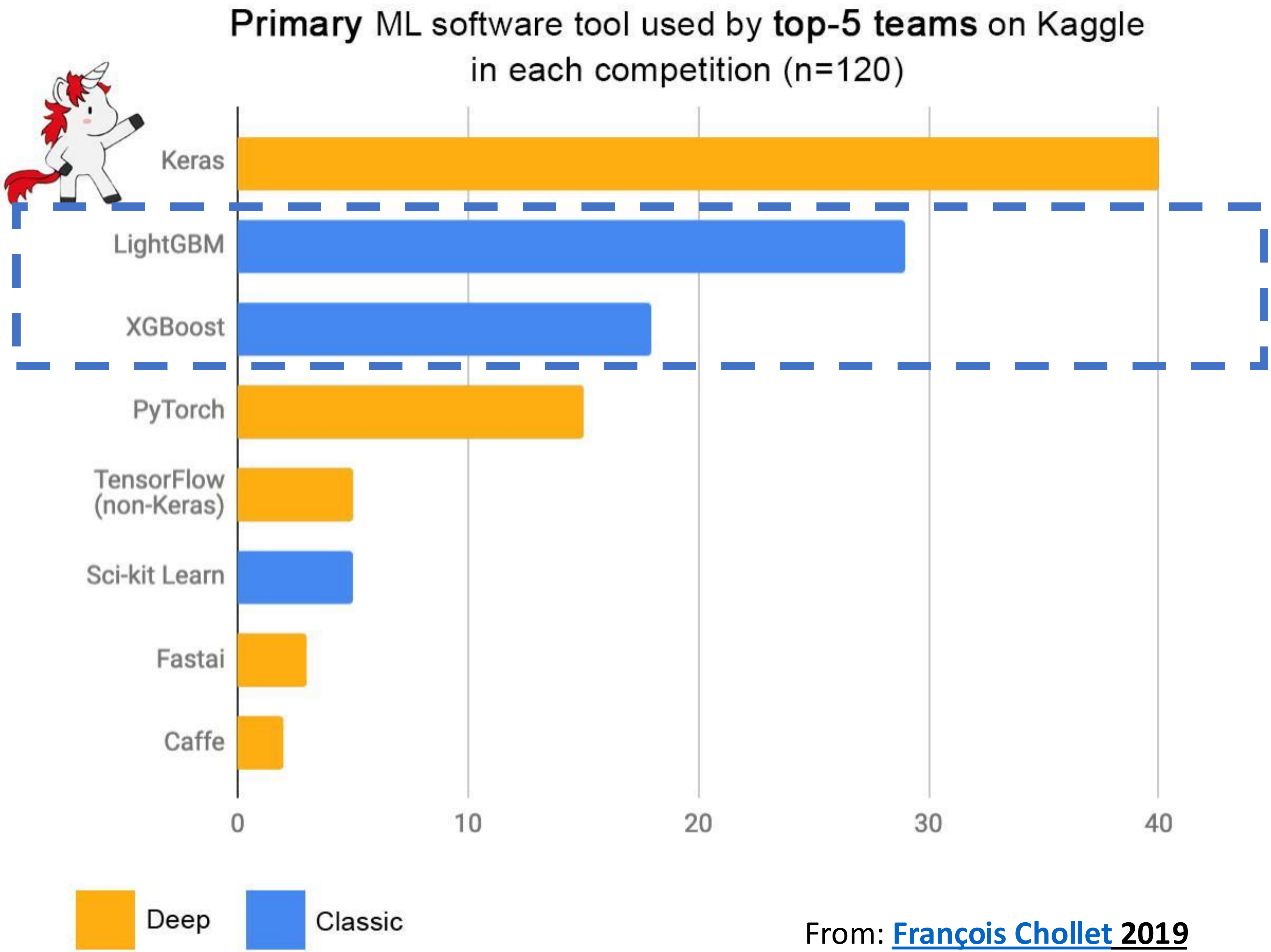
  $O(n)$, α_i $i=1,\cdots,n$

  ✓ While we have a lot of inputs, at the end we only care about the support vectors and these are usually a small group of samples

  ✓ The maximizing of the margin acts as a sort of regularization term leading to reduced overfitting

L22: Decision Tree / Bagged DT/ Random Forest

Data	Tabular
Task	Classification / Regression
Representation	Partition feature space into set of rectangles, local smoothness
Score Function	Greedy to find partitions
Search/Optimization	Split with Purity measure / e.g. IG / cross-entropy / Gini /
Models, Parameters	Tree Model (s), i.e. space partition



From: [François Chollet](#) 2019

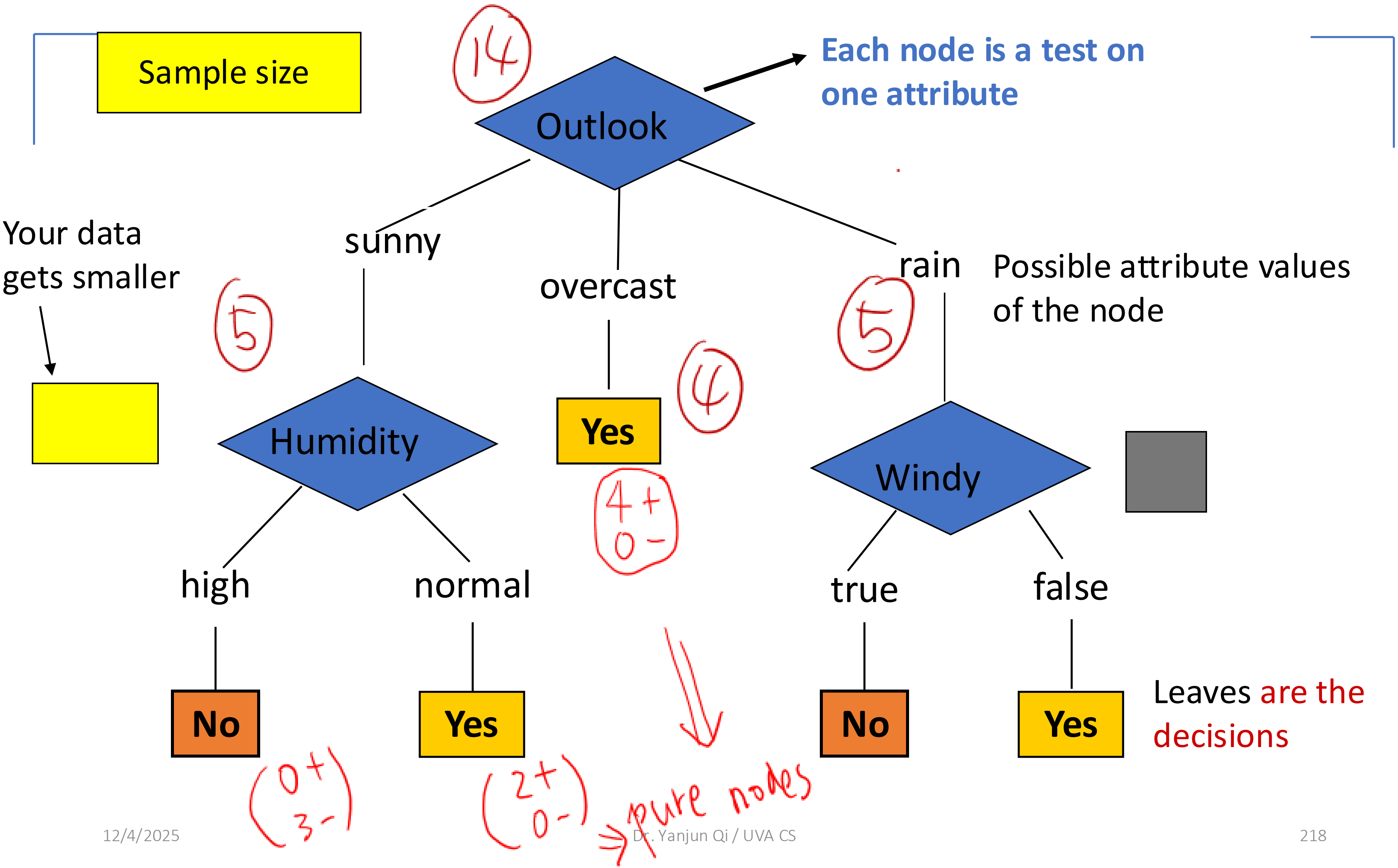
Example

- Example: Play Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

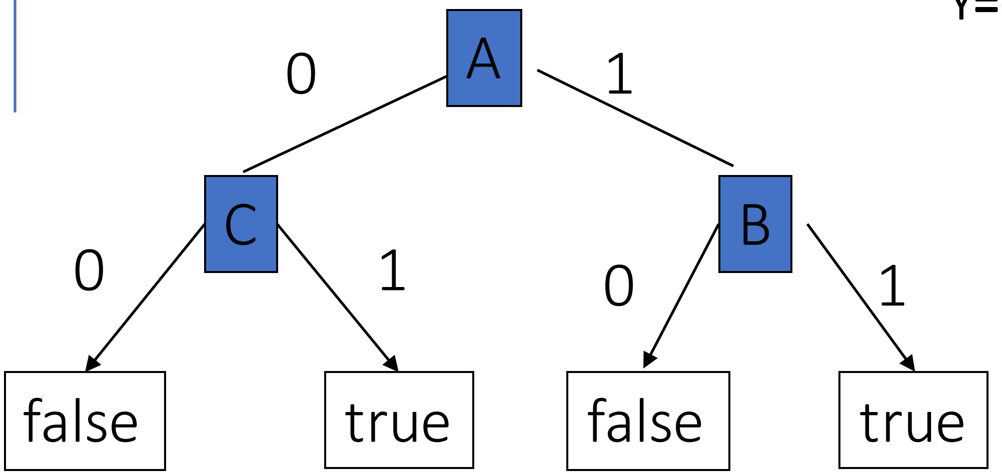
Anatomy of a decision tree

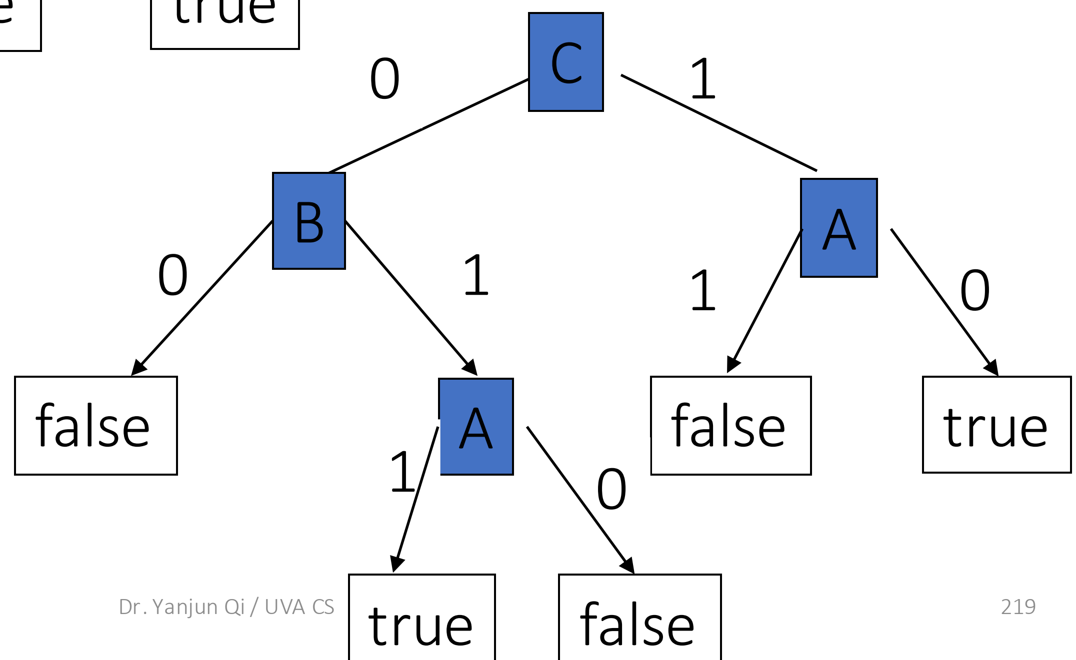


Same concept / different representation

Y=((A and B) or ((not A) and C))

not unique

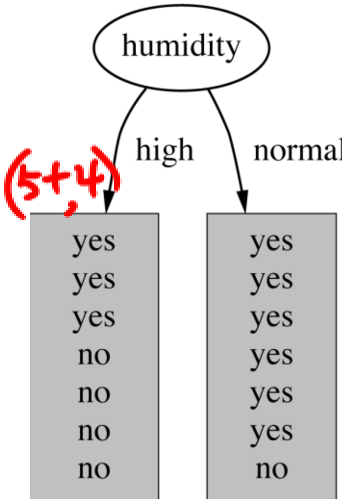




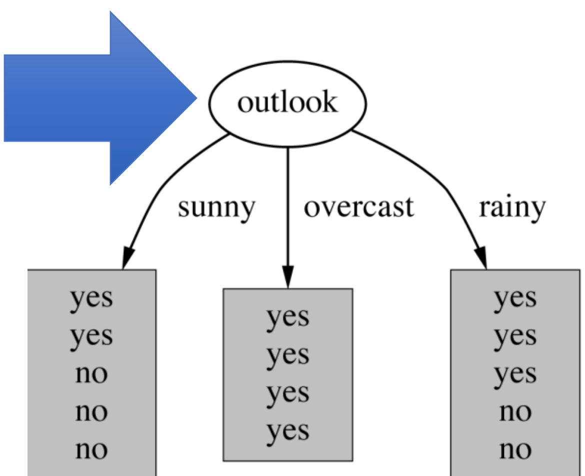
How do we choose which attribute to split ?

Which attribute should be used first to test?

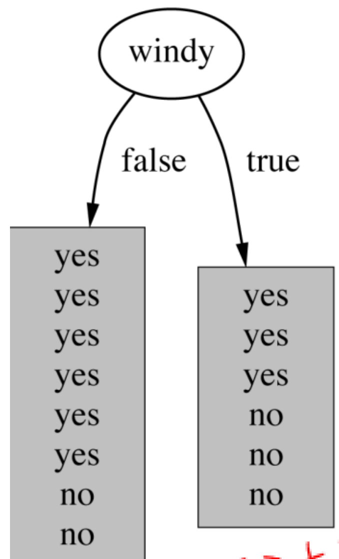
Intuitively, you would prefer the one that *separates* the training examples as much as possible. wrt. class distribution



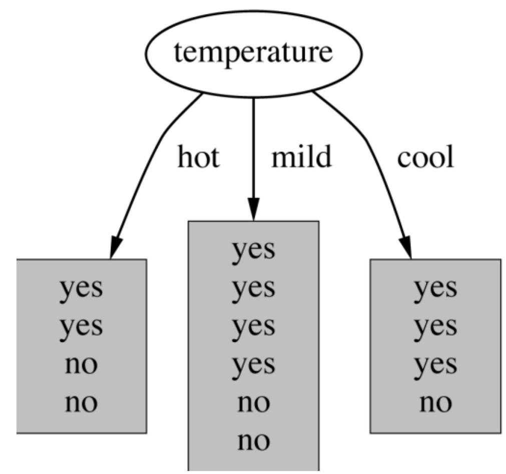
(5+,4-) (6+,1-)



(2+ 3-) (4+ 0-) pure nodes (3+ 2-)

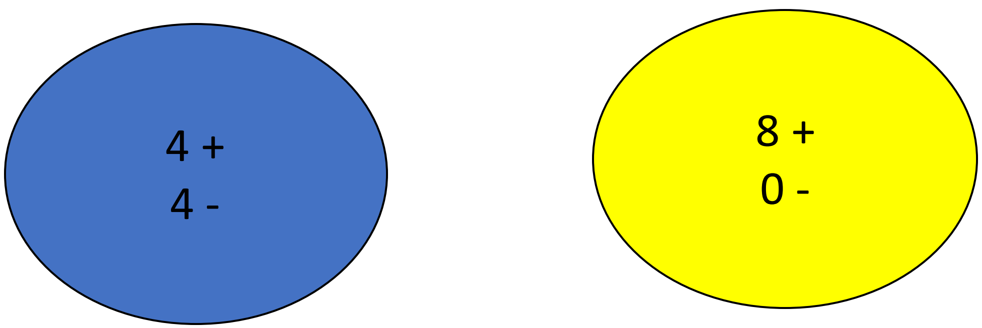


(6+ 2-) (3+ 3-)



Entropy Lower → better purity

- Entropy measures the purity



The distribution is less uniform
Entropy is lower
The node is purer

Information gain

- IG(X,Y)=H(Y)-H(Y|X)

Reduction in uncertainty of Y by knowing a feature variable X

Information gain:

= (information before split) – (information after split)

= entropy(parent) – [average entropy(children)]

Fixed − the lower, the better (children nodes are purer) = For IG, the higher, the better

Information gain

- IG(X,Y)=H(Y)-H(Y|X)

$$\Rightarrow \text{pick } x_i \text{ BY } \underset{x_i}{\operatorname{argmax}} \begin{cases} IG(X_1, Y) \\ IG(X_2, Y) \\ IG(X_m, Y) \end{cases}$$

Reduction in uncertainty of Y by knowing a feature variable X

Information gain:

= (information before split) – (information after split)

= entropy(parent) – [average entropy(children)]

$H(Y)$

Fixed

–

$H(Y|X_i)$

the lower, the better (children nodes are purer)

=

For IG, the higher, the better

Conditional entropy

$$H(Y) = -\sum_{i} p(y_i)\log_2 p(y_i)$$

$$H(Y \mid X = x_j) = -\sum_{i} p(y_i \mid x_j)\log_2 p(y_i \mid x_j)$$

$$H(Y \mid X) = \sum_{j} p(x_j) H(Y \mid X = x_j)$$

$$= -\sum_{j} p(x_j) \sum_{i} p(y_i \mid x_j)\log_2 p(y_i \mid x_j)$$

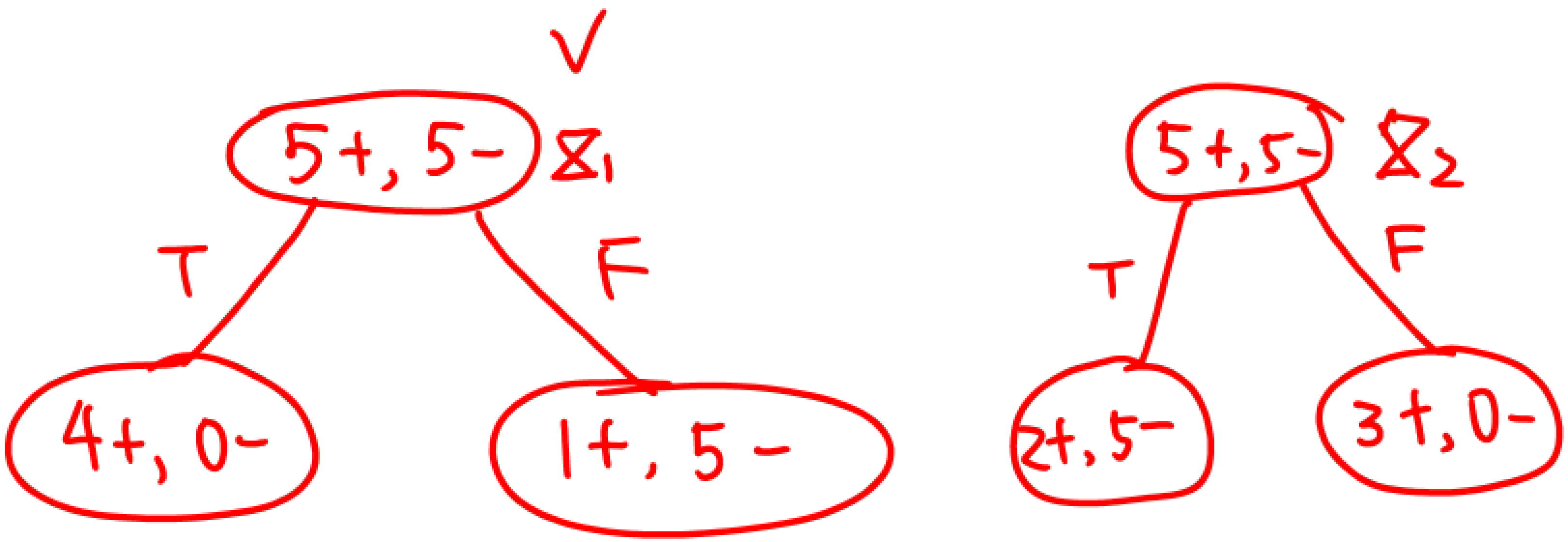
Example

Attributes Labels

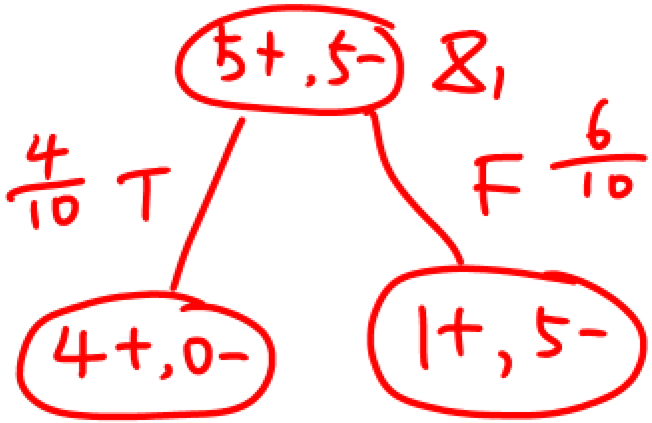
X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Which one do we choose

X1 or X2?



X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1



$$H(Y|X_1=T) = -\{p(Y=+|X_1=T)\log p(Y=+|X_1=T)$$

4+, 0- $\Rightarrow$

$$+\; p(Y=-|X_1=T)\log p(Y=-|X_1=T)\}$$

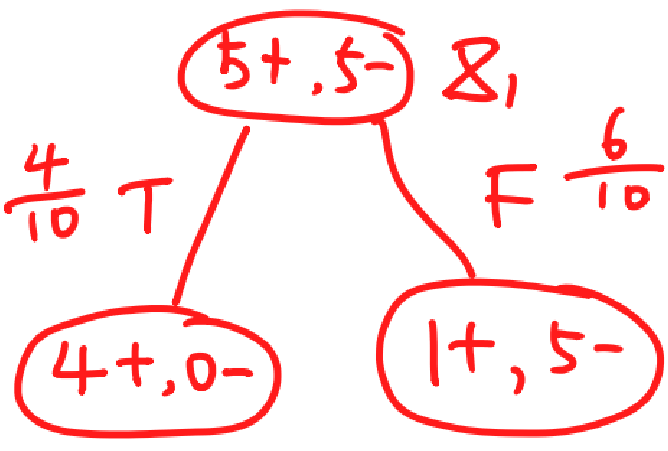
$$= 0$$

$$H(Y|X_1=T) = \begin{matrix}4+\\0-\end{matrix} \Rightarrow -\left(P(+)\log P(+) + P(-)\log(P(-))\right)$$

$$= -(1\log_2 1 + 0\log 0) = 0$$

$$H(Y|X_1=F) = \begin{matrix}1+\\5-\end{matrix} \Rightarrow -\left(P(+)\log P(+) + P(-)\log P(-)\right)$$

$$= -\left(\frac{1}{6}\log\frac{1}{6} + \frac{5}{6}\log\frac{5}{6}\right)$$



$$H(Y|X_1) = \frac{4}{10}H(Y|X_1=T) + \frac{6}{10}H(Y|X_1=F)$$

Intuition of Node Splitting

More certain children nodes

More information

Less entropy

Lower value conditional entropy

Larger information gain

Entropy

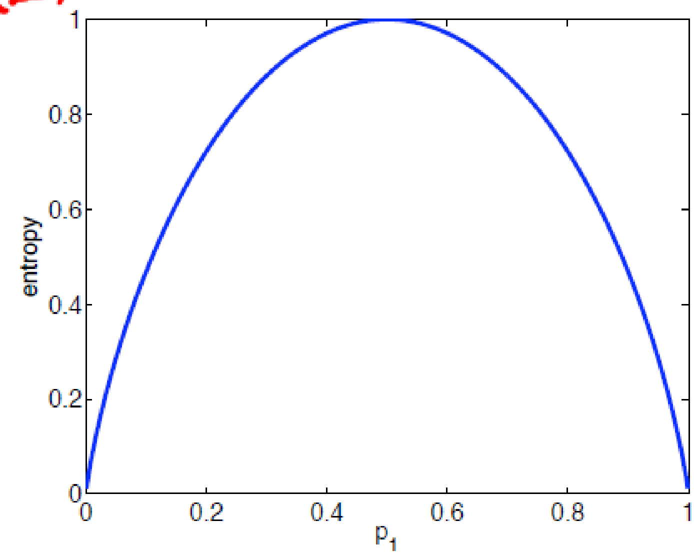
- If there are k possible outcomes

$$H(X) \le \log_2 k$$

- Equality holds when all outcomes are equally likely

- The more the probability distribution that deviates from uniformity, the lower the entropy

H(X)



p(H)

$$H(X) = E(I(X)) = \sum_i p(x_i) I(x_i) = -\sum_i p(x_i) \log_2 p(x_i)$$

e.g. for a random binary variable

Many tree building algorithms...(EXTRA)

Feature	C4.5	CART	CHAID	CRUISE	GUIDE	QUEST
Unbiased Splits				√	√	√
Split Type	*u*	*u,l*	*u*	*u,l*	*u,l*	*u,l*
Branches/Split	≥2	2	≥2	≥2	2	2
Interaction Tests				√	√	
Pruning	√	√		√	√	√
User-specified Costs		√	√	√	√	√
User-specified Priors		√		√	√	√
Variable Ranking		√			√	
Node Models	*c*	*c*	*c*	*c,d*	*c,k,n*	*c*
Bagging & Ensembles					√	
Missing Values	*w*	*s*	*b*	*i,s*	*m*	*i*

b, missing value branch; *c*, constant model; *d*, discriminant model; *i*, missing value imputation; *k*, kernel density model; *l*, linear splits; *m*, missing value category; *n*, nearest neighbor model; *u*, univariate splits; *s*, surrogate splits; *w*, probability weights

From Dr. Camilo Fosco

11/18

Today Roadmap

- 0. To Remind:
  - Dec. 9th Final Exam in Person (notes only!)
  - Next week: Nov.25th class zoom (100%! Based on Survey 3)
  - Please sign up your project final presentation time slot ASAP!
  - HW5 in Canvas! Project Due entry in Canvas!

- 1. Review some course content
  - A. Hier. Clustering (an example to run through)
  - B. Boosting
  - C. Decision Tree (how to calculate IG, an example to run through!)

- 2. Quiz 12
  - FYI: Q11 is the hardest over all past quizzes … So no worries if not full scored
  - Makeup quizzes Q14 + Q15 in the week of Dec. 2nd
  - We will use your top10 quiz grade … only top 10 !!

S4: Lecture 23 : : **Ensemble**

- Framework of Ensemble:

  - 1. Get a set of classifiers $f_1(x), f_2(x), f_3(x), \ldots\ldots$

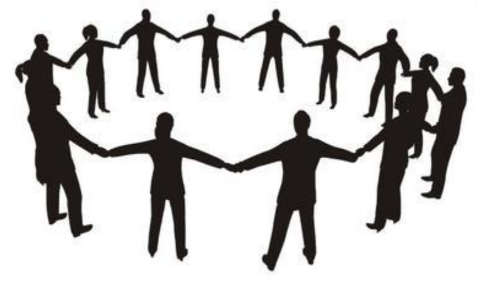
$f_B(x)$

They should be diverse.

How to have different training data sets
- Re-sampling your training data to form a new set
- Re-weighting your training data to form a new set

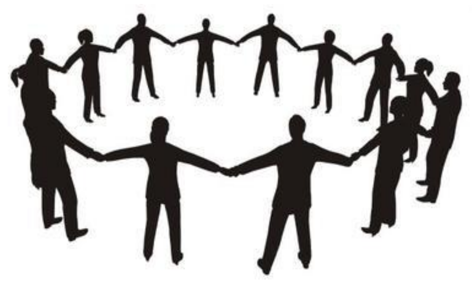
blend

  - 2. Aggregate the classifiers (*properly*)

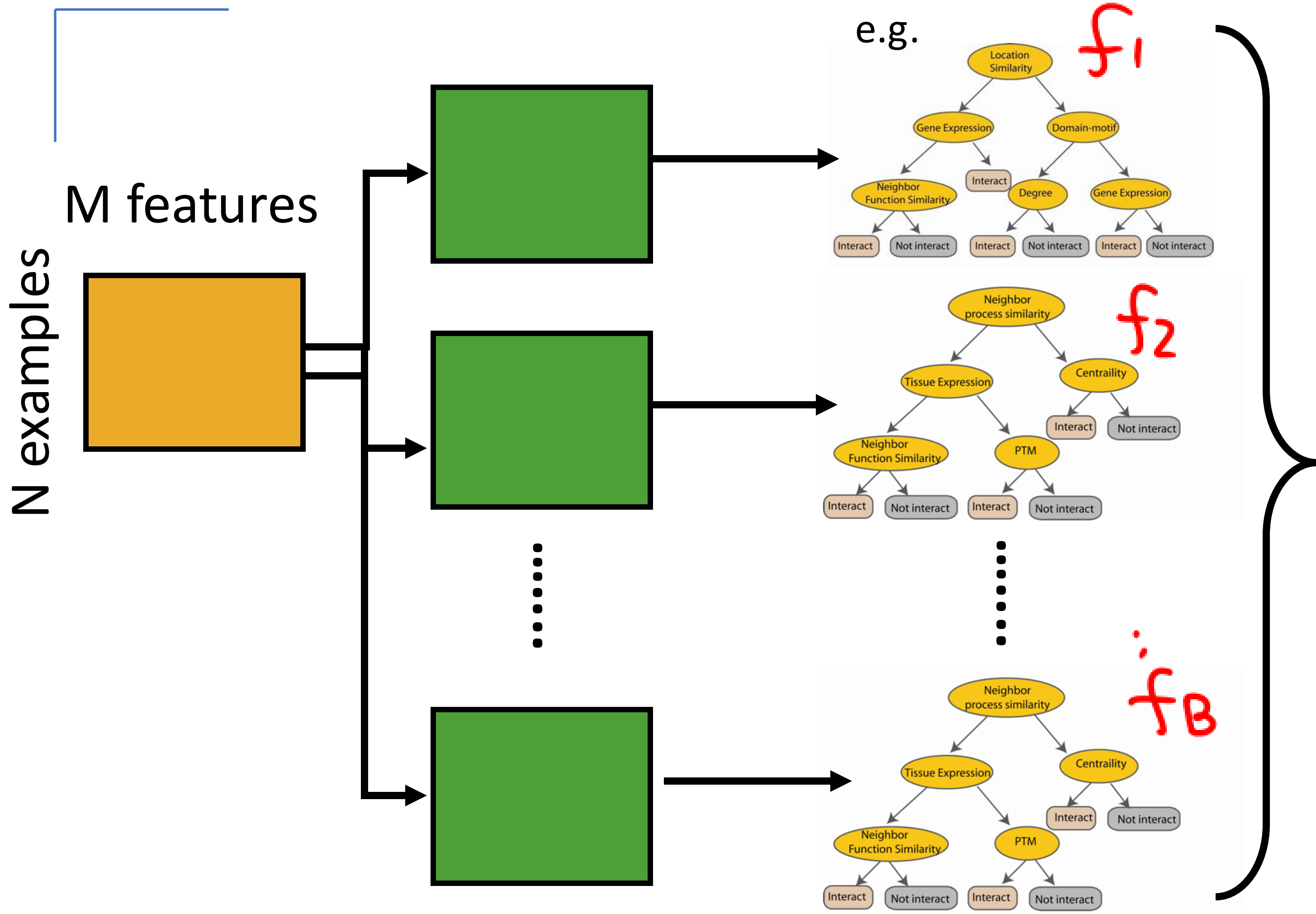


S4: Lecture 23 :
Random Forest / Boosting : **Ensemble**

- Bagging / reduce variance
  - Bagged Decision Tree
  - Random forests:
- Boosting
  - Adaboost / reduce bias, reduce variance
  - Xgboost
- Stacking / Gradient Boosting



Bagging of DT **Classifiers**



$$f_E = \frac{1}{B}\sum_{i=1}^{B} f_i$$

Take the majority vote

i.e. Refit the model to each bootstrap dataset, and then examine the behavior over the B replications.

With vs Without Replacement

- Bootstrap with replacement can keep the sampling size the same as the original size for every repeated sampling. The sampled data groups are independent on each other.

- Bootstrap without replacement cannot keep the sampling size the same as the original size for every repeated sampling. The sampled data groups are dependent on each other.

$$Var(f_i) \quad vs. \quad Var\left(\frac{1}{B}\sum_{i=1}^{B} f_i\right)$$

Decision Boundary Comparison

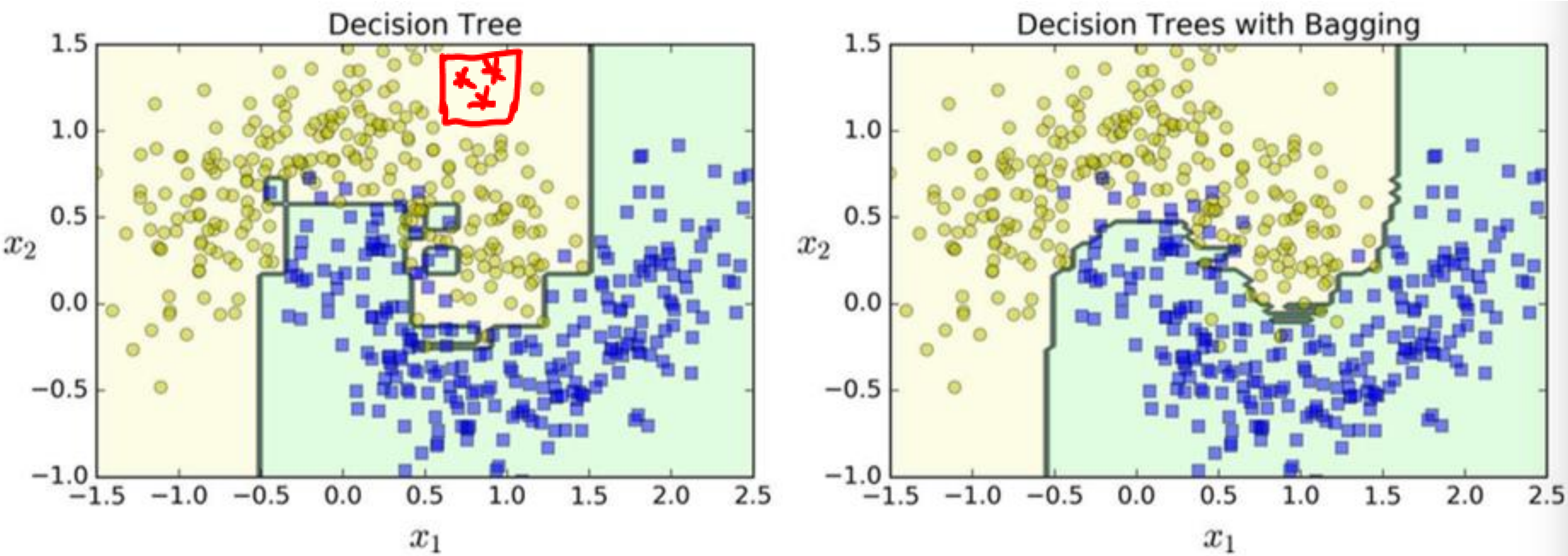
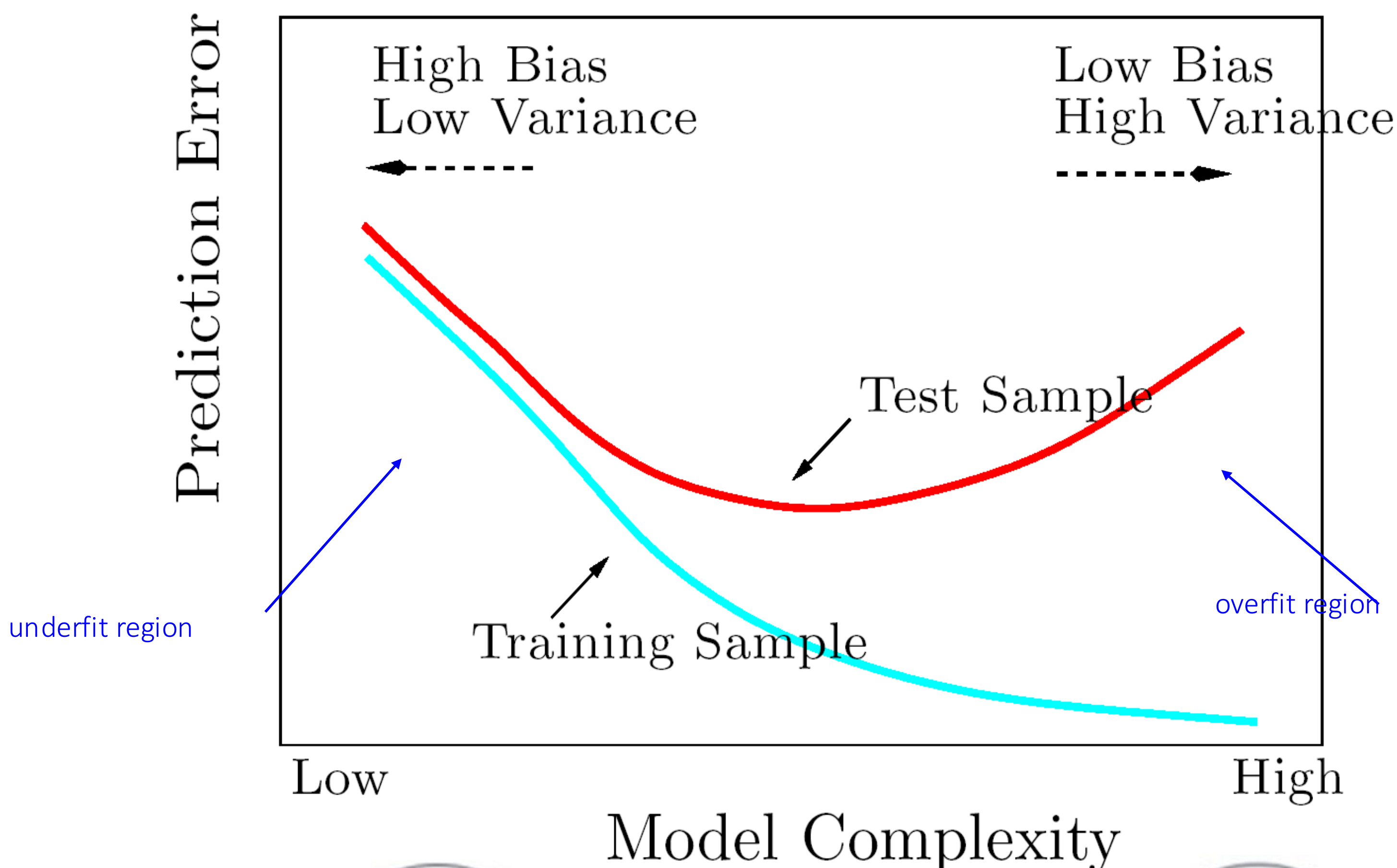


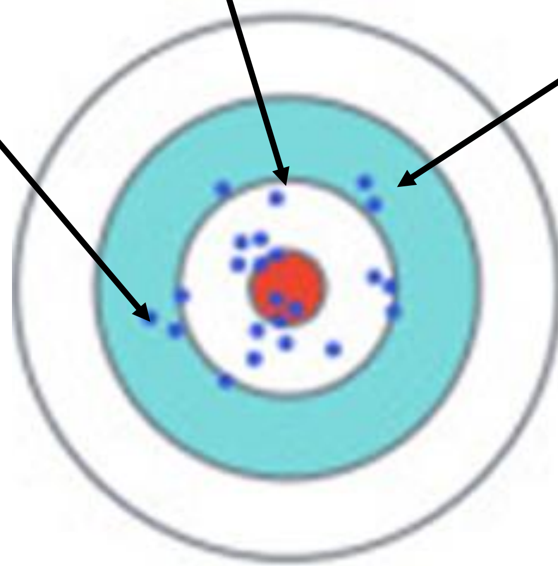
Figure 7-5. A single Decision Tree versus a bagging ensemble of 500 trees

Recap: Bias-Variance Tradeoff / Model Selection



classifiers $f_1(x)$, $f_2(x)$, $f_3(x)$, ... $f_B(x)$

S_1 S_2 S_B



A complex model will have large variance.

We can average complex models to reduce variance.

If we average all the f_i, is it close to f^*

$$E[\hat{f}] = f^*$$

Bagged Trees ➔ Random Forests

1. Construct subset $(x_1^*, y_1^*), \ldots, (x_n^*, y_n^*)$ by sampling original training set with replacement.
2. Build tree-structured learners $h(x, \Theta_k)$, where at each node, m predictors at random are selected before finding the best split.
   - Gini Criterion.
   - No pruning.
3. Combine the predictions (average or majority vote) to get the final result.

Why correlated trees (in Bagging) are not ideal ?

Assuming each tree has variance σ^2

If simply identically distributed, then average variance is

$\rho_{ij} = \mathrm{Corr}(f_i, f_j) = \rho$

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

As $B \to \infty$, second term $\to 0$

Thus, the pairwise correlation always affects the variance

Why correlated trees (in Bagging) are not ideal ?

How to deal?

If we reduce *m* (the number of dimensions we actually consider in each splitting),
then we reduce the pairwise tree correlation

Thus, variance will be reduced.

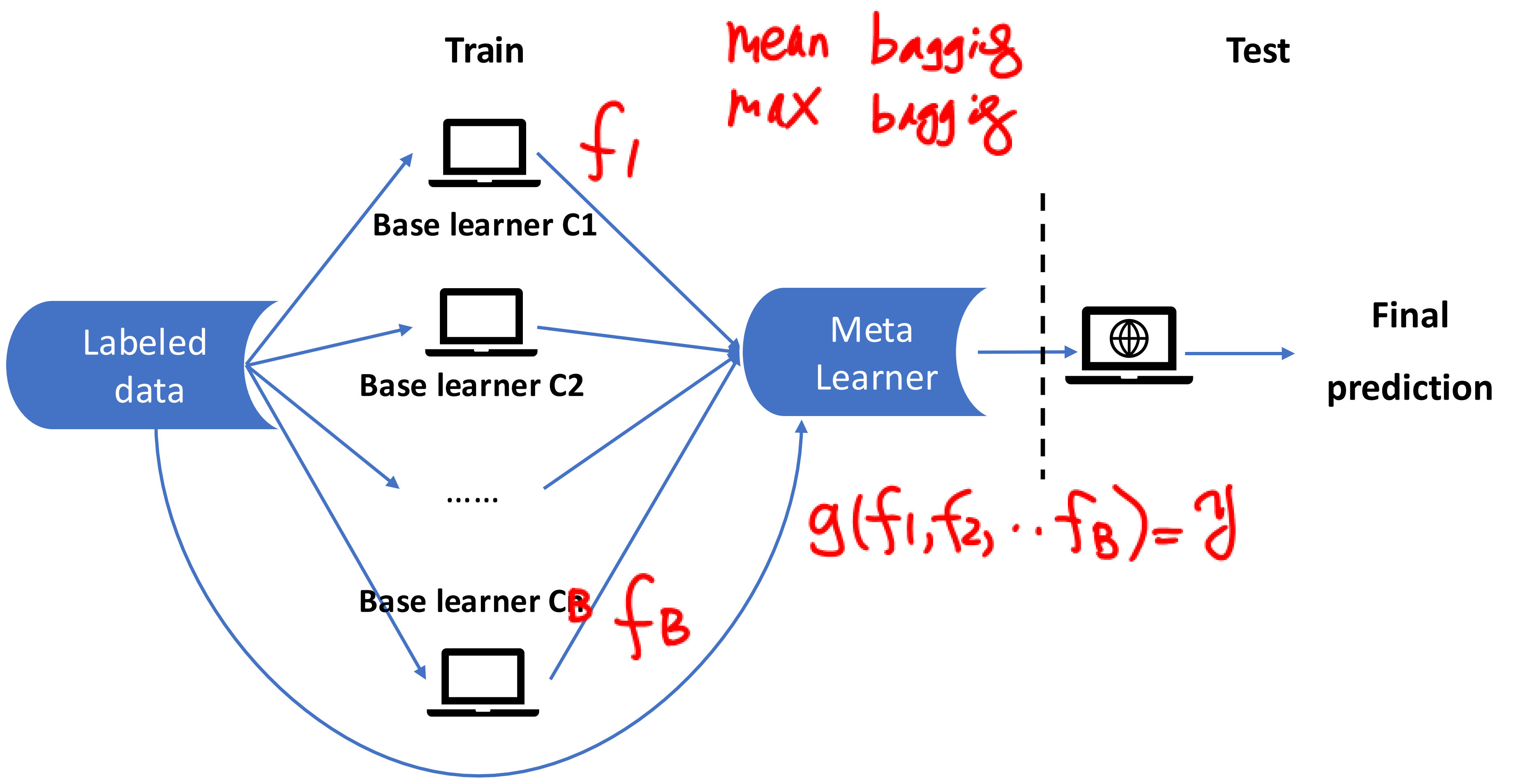
S4: Lecture 23 :

Random Forest / Boosting : **Ensemble**

- Bagging / reduce variance
  - Bagged Decision Tree
  - Random forests:
- Boosting
  - Adaboost / reduce bias, reduce variance
  - Xgboost
- Stacking / Gradient Boosting

Stacking

- Main Idea: Learn and combine multiple classifiers



Generating Base and Meta Learners

- Base model—efficiency, accuracy and diversity
  - Sampling training examples
  - Sampling features
  - Using different learning models

- Meta learner
  - Majority voting
  - Weighted averaging
  - .....

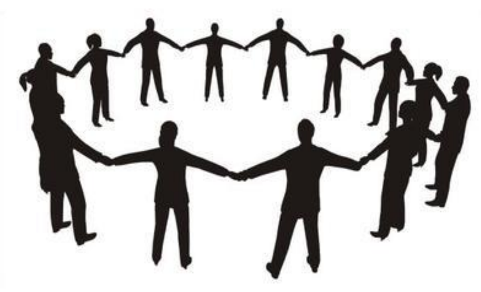
  Unsupervised
  - Higher level classifier — Supervised (e.g. Xgboost as blender)

From Dr. Rich UVa

S4: Lecture 23 :

Random Forest / Boosting : **Ensemble**

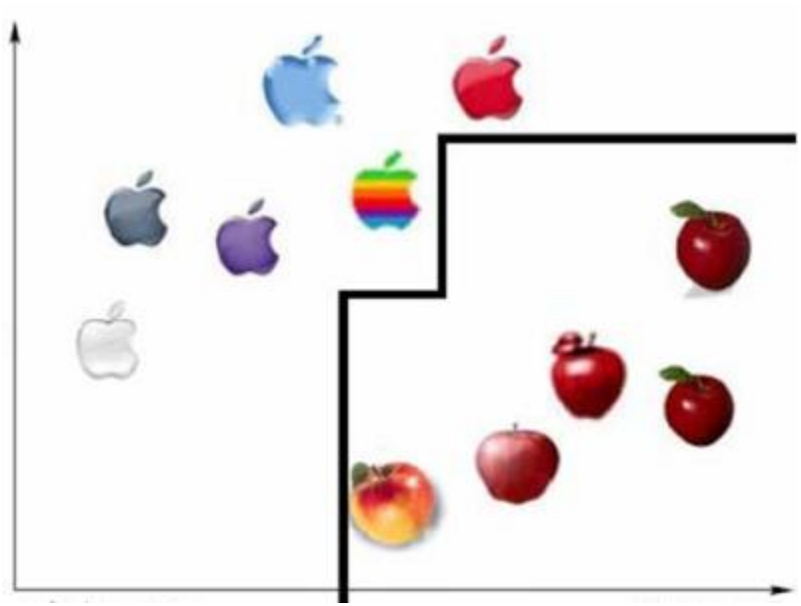
- Bagging / reduce variance
  - Bagged Decision Tree
  - Random forests:
- Boosting
  - Adaboost / reduce bias, reduce variance
  - Xgboost
- Stacking / Gradient Boosting



Boosting Strategies

1. Have many rules (base classifiers) to **vote** on the decision
    1. Base learners are shallow decision trees!!!
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules

Final Classifier is the additive combination of base rules:



From Dr. Rich UVa

Boosting vs. Bagging

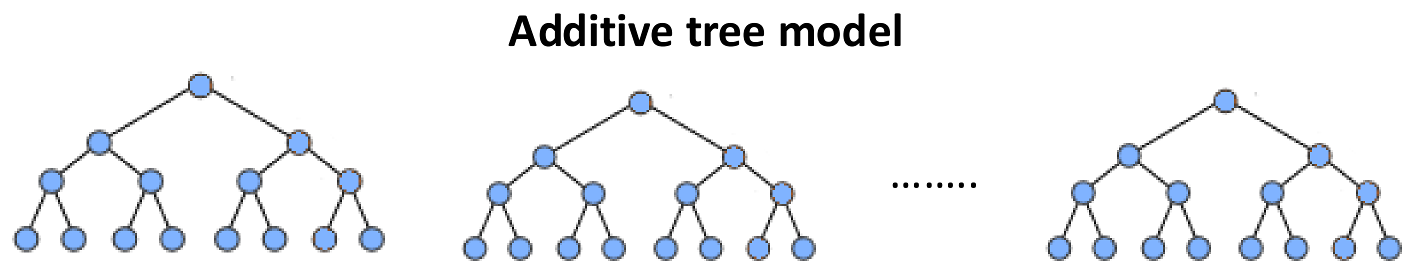
- Similar to bagging, boosting combines a weighted sum of many classifiers, thus **both reduce variance**.

- One key difference: unlike bagging, boosting fit the tree to the entire training set, and adaptively weight the examples.

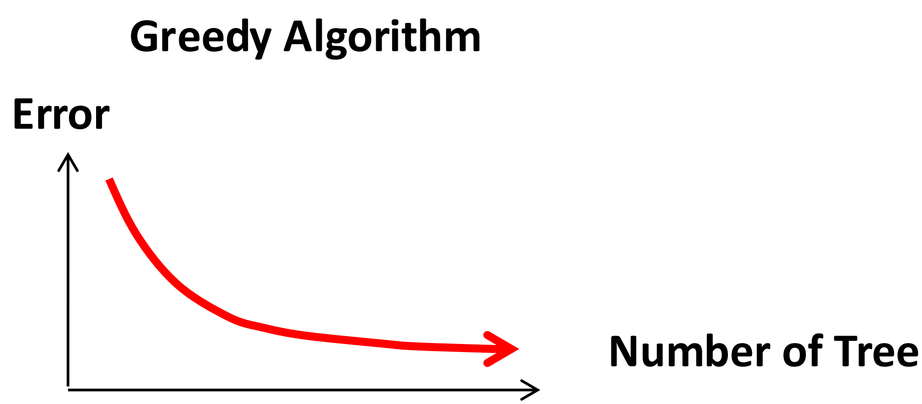
- Boosting tries to do better at each iteration, (by making model a bit more complex), thus **it reduces bias**.

XGBoost

- Additive tree model: add new trees that complement the already-built ones
- Response is the optimal linear combination of all decision trees
- Popular in Kaggle Competitions for efficiency and accuracy



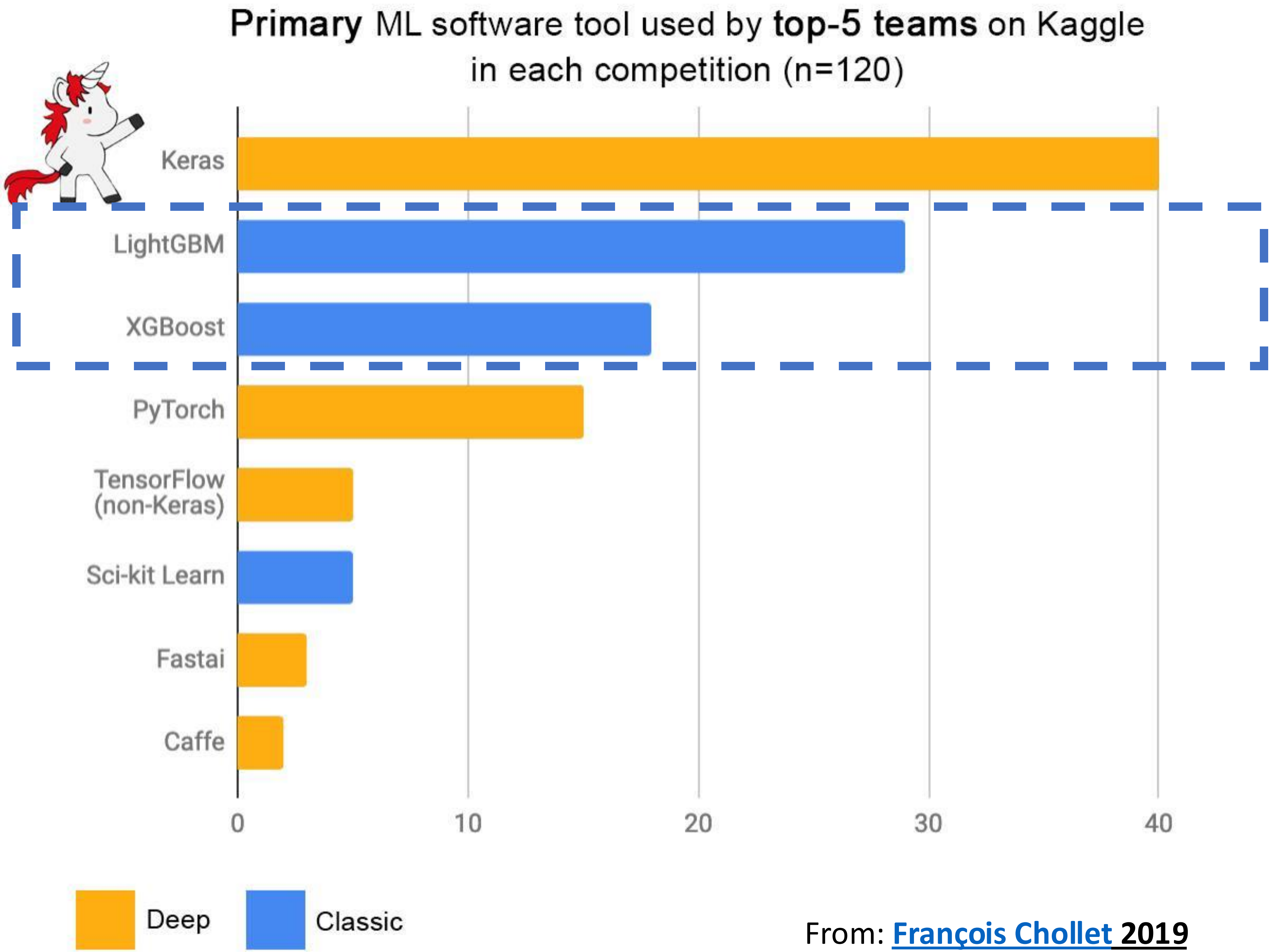
More in 18c-extraBoosting Slides



XGBoost Implementation

- XGBoost is a very efficient Gradient Boosting Decision Tree implementation with some interesting features:
- **Regularization:** Can use L1 or L2 regularization.
- **Handling sparse data:** Incorporates a sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data.
- **Weighted quantile sketch:** Uses distributed weighted quantile sketch algorithm to effectively handle weighted data.
- **Block structure for parallel learning:** Makes use of multiple cores on the CPU, possible because of a block structure in its system design. Block structure enables the data layout to be reused.
- **Cache awareness:** Allocates internal buffers in each thread, where the gradient statistics can be stored.
- **Out-of-core computing:** Optimizes the available disk space and maximizes its usage when handling huge datasets that do not fit into memory.

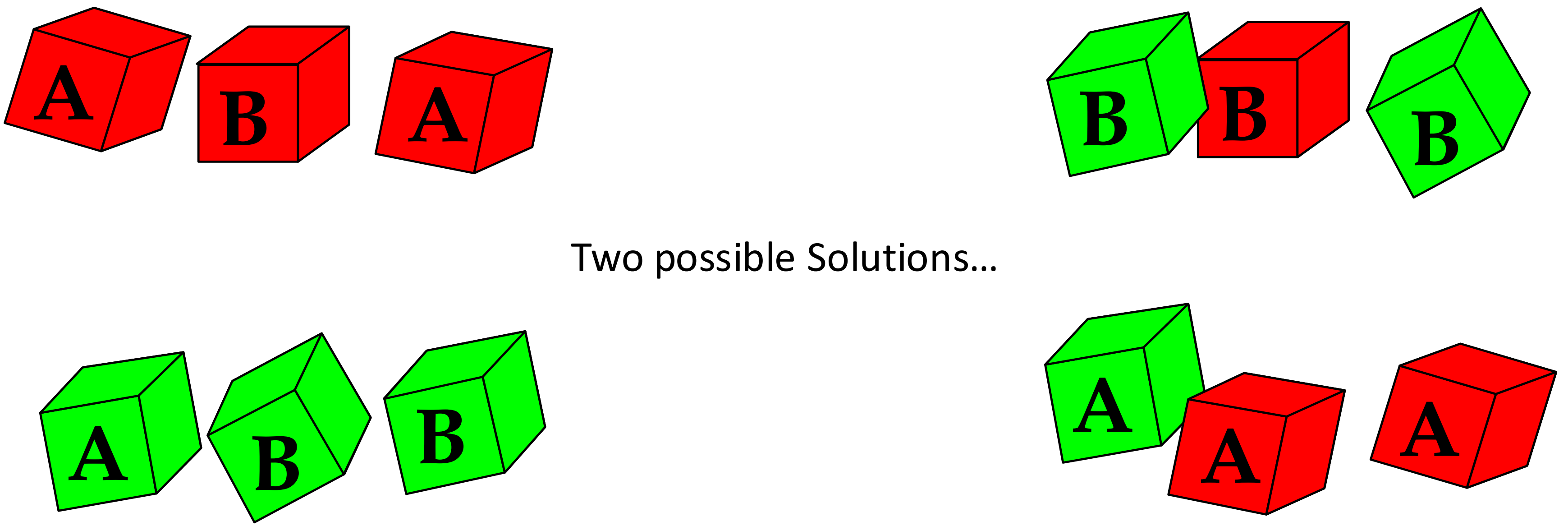
credit: Camilo Fosco



From: François Chollet 2019

S5: Lecture 24:
Unsupervised Clustering (I): Hierarchical

example: clustering is subjective



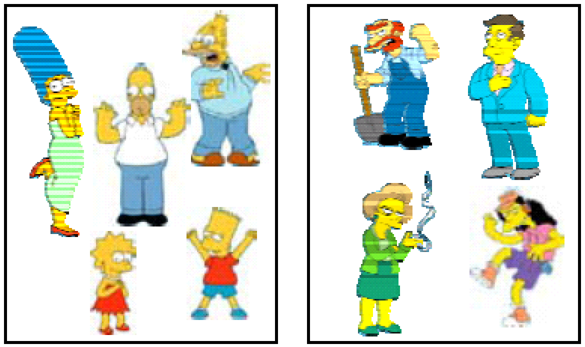
Two possible Solutions…

Course Content Regarding Tasks

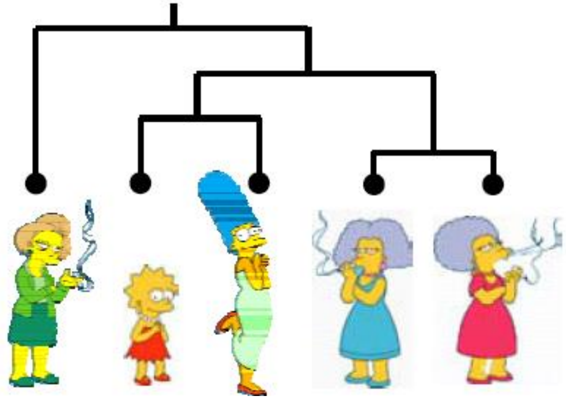
- ~~Regression (supervised)~~ — Y is a continuous
- ~~Learning theory~~ — About f()
- ~~Classification (supervised)~~ — Y is a discrete
- Unsupervised models — NO Y
- ~~Graphical models~~ — About interactions among Y,X1,. Xp
- Reinforcement Learning — Learn to Interact with environment

Clustering Algorithms

- Partitional algorithms
  - Usually start with a random (partial) partitioning
  - Refine it iteratively
    - K means clustering
    - Mixture-Model based clustering



- Hierarchical algorithms
  - Bottom-up, agglomerative
  - Top-down, divisive



(How-to) Hierarchical Clustering

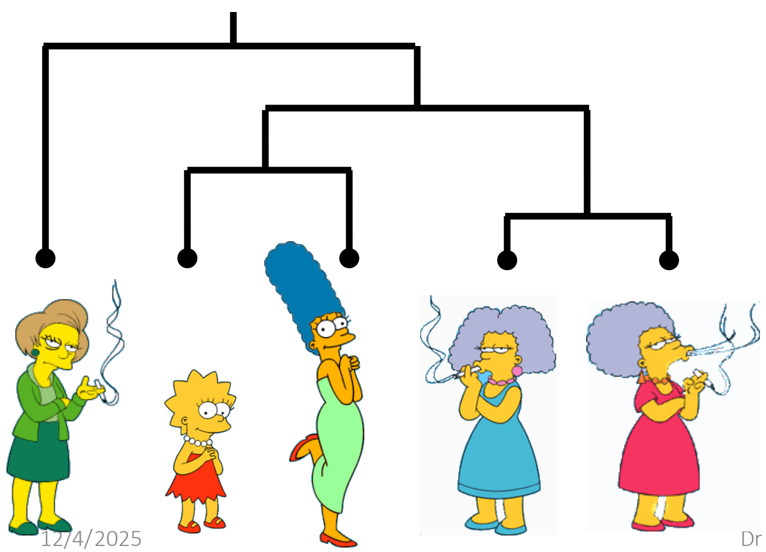
- Given: a set of objects and the pairwise distance matrix

- Find: a tree that optimally hierarchical clustering objects?
  - Globally optimal: exhaustively enumerate all tree
  - Effective heuristic methods:

(How-to) Hierarchical Clustering

The number of dendrograms with n leafs
$= (2n-3)!/[(2^{(n-2)})\ (n-2)!]$

Number of Leafs	Number of Possible Dendrograms
2	1
3	3
4	15
5	105
...	...
10	34,459,425



Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

A greedy local optimal solution

Clustering: the process of grouping a set of objects into classes of similar objects ➔
- high intra-class similarity
- low inter-class similarity

How to decide the distances between clusters ?

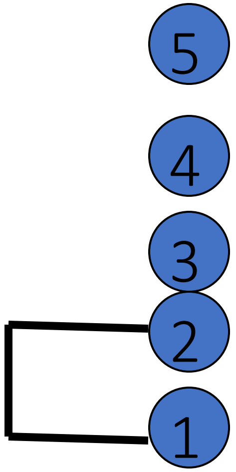
- Single-Link
  - Nearest Neighbor: their closest members.
- Complete-Link
  - Furthest Neighbor: their furthest members.
- Average:
  - average of all cross-cluster pairs.

Summary of Hierarchal Clustering Methods

- No need to specify the number of clusters in advance.
- Hierarchical structure maps nicely onto human intuition for some domains
- They do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects.
- Like any heuristic search algorithms, local optima are a problem.
- Interpretation of results is (very) subjective.

Example: single link

$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & & & & \\
2 & 0 & & & \\
6 & 3 & 0 & & \\
10 & 9 & 7 & 0 & \\
9 & 8 & 5 & 4 & 0
\end{bmatrix}
\end{array}
$$



11/25

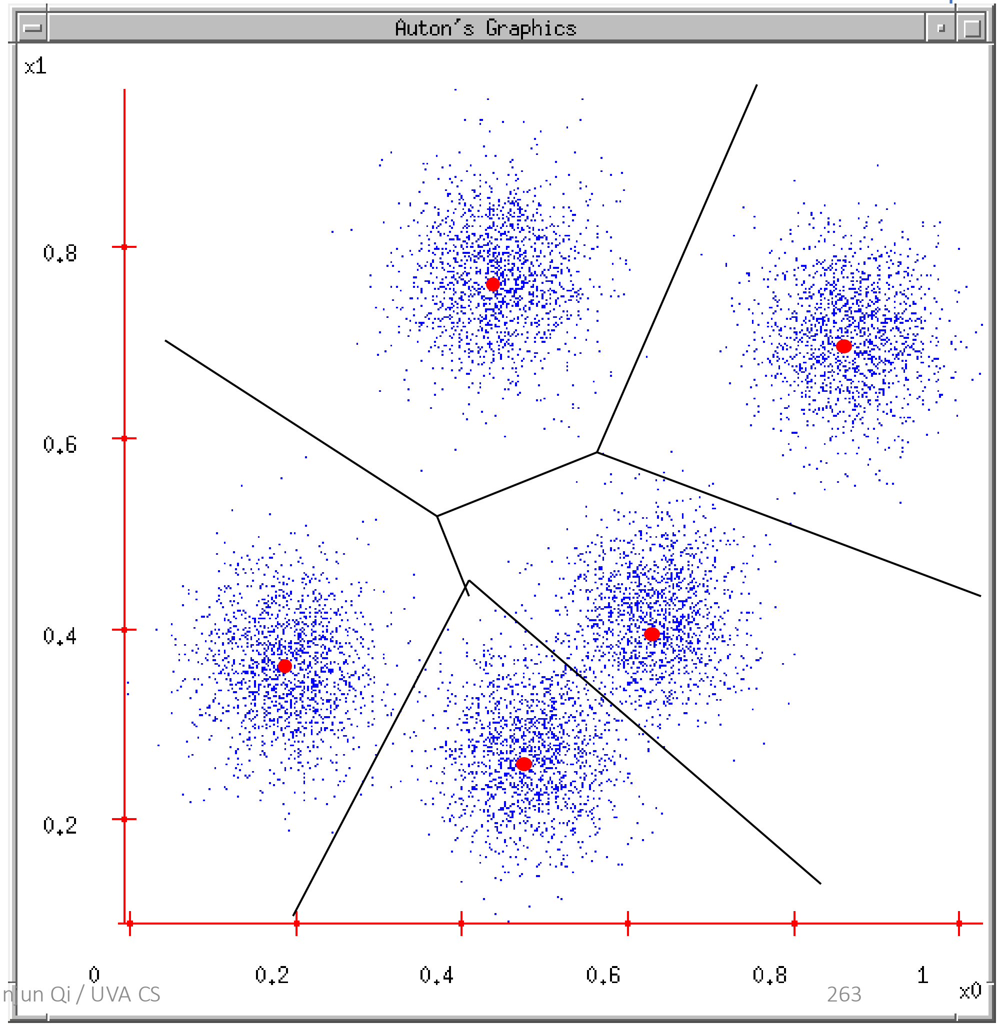
Today's Roadmap

- 0. Logistics:
  - Dec. 9th Final Exam in Person (notes only!)
  - Please sign up for your project final presentation time slot ASAP!
  - HW5 Due this weekend
  - We will use your top10 quiz grades … only top 10 !!
- 1. Review
  - A. Going over HW4 solutions and code
  - B. K-means quick review
  - C. Then Quiz 13
  - D. second Half of today's time will be the "shark tank" sessions
- 2. Next week:
  - a. Makeup quizzes Q14 + Q15 in the week of Dec. 2nd
  - b. Next week: for both sessions Dec 2 + Dec 4 (waiting for 18 votes!)
  - C. RL + RL Gym (not part of final exam!) --- next Tuesday
  - D. Thorough review sessions next Tue + Thursday

K-means

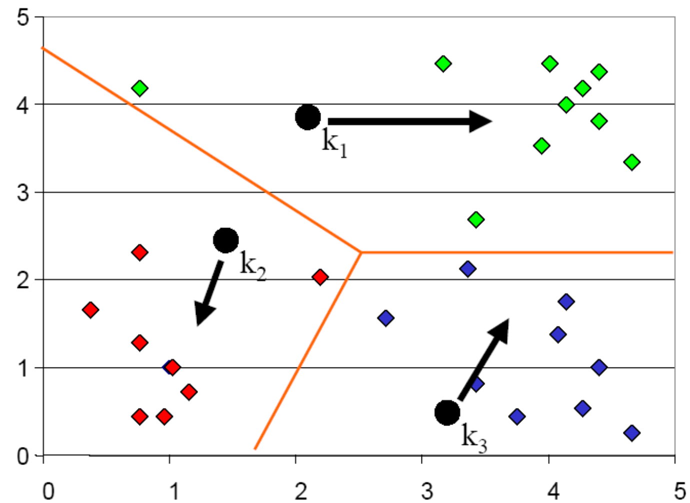
1. Ask user how many clusters they'd like. (e.g. k=5)
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns

Any Computational Problem?



Seed Choice

- Results can vary based on random seed selection.



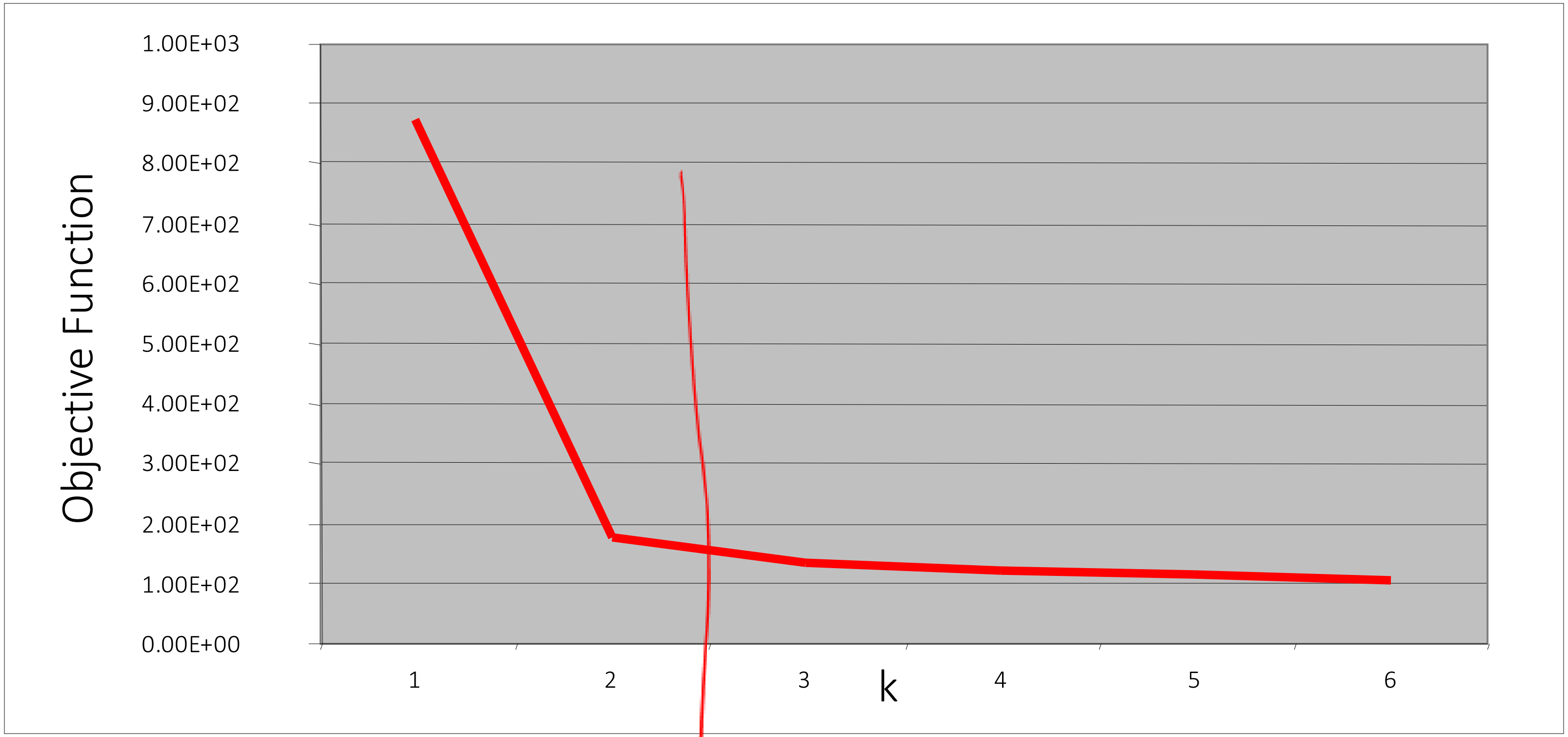
K

$C_1, C_2, \ldots, C_K$

- Some seeds can result in poor convergence rate, or convergence to sub-optimal clustering.
  - Select good seeds using a heuristic (e.g., sample least similar to any existing mean)
  - Try out multiple starting points (very important!!!)
  - Initialize with the results of another method.

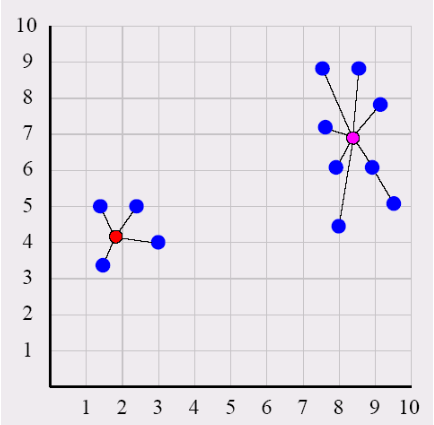
We can plot the objective function values for k equals 1 to 6…

The abrupt change at k = 2, is highly suggestive of two clusters in the data. This technique for determining the number of clusters is known as "knee finding" or "elbow finding".



Note that the results are not always as clear cut as in this toy example

K-Means Clustering : Convergence



- Why should the K-means algorithm ever reach a fixed point?
  - A state in which clusters don't change.

- K-means is a special case of a general procedure known as the Expectation Maximization (EM) algorithm.
  - EM is known to converge.
  - Number of iterations could be large.

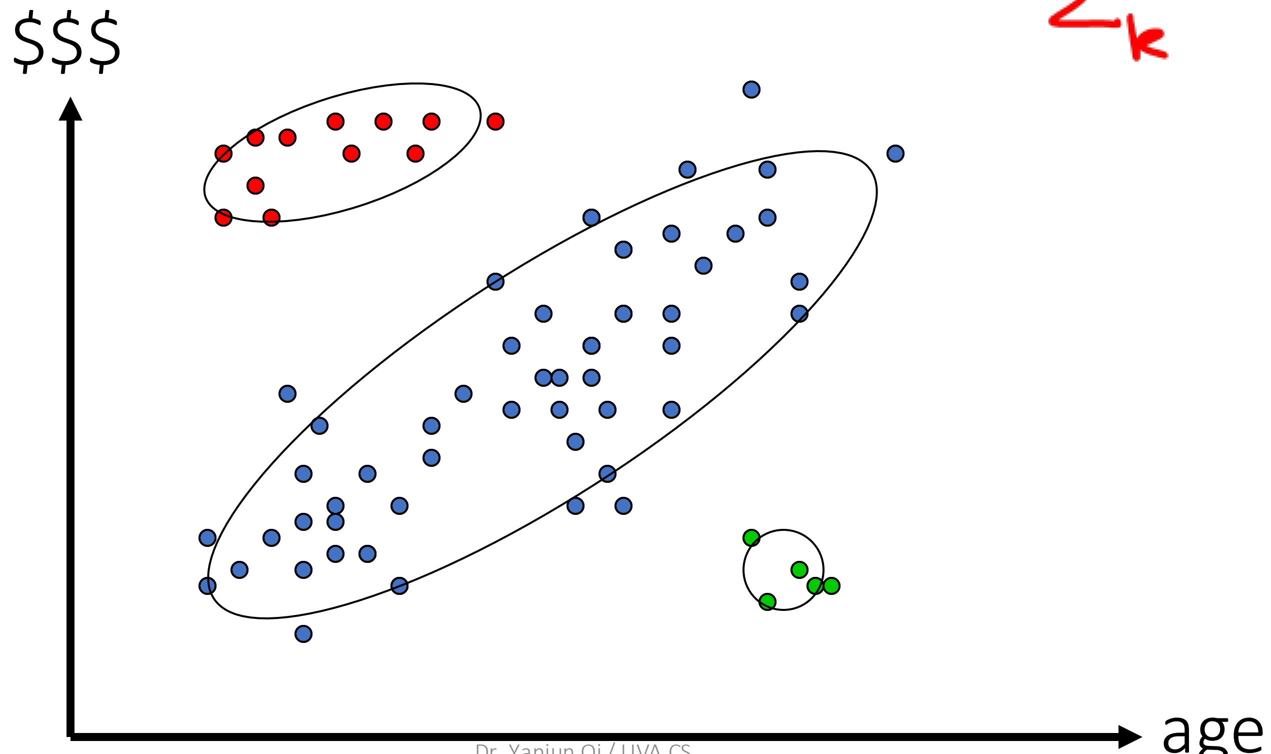
- Optimize the goodness measure (i.e., minimize the Loss function)
  - sum of squared distances from cluster centroid:
- Reassignment monotonically decreases the goodness measure since each vector is assigned to the closest centroid.

Partitional clustering (e.g. K=3)

$k=1,2,..K$

μ_k

Σ_k

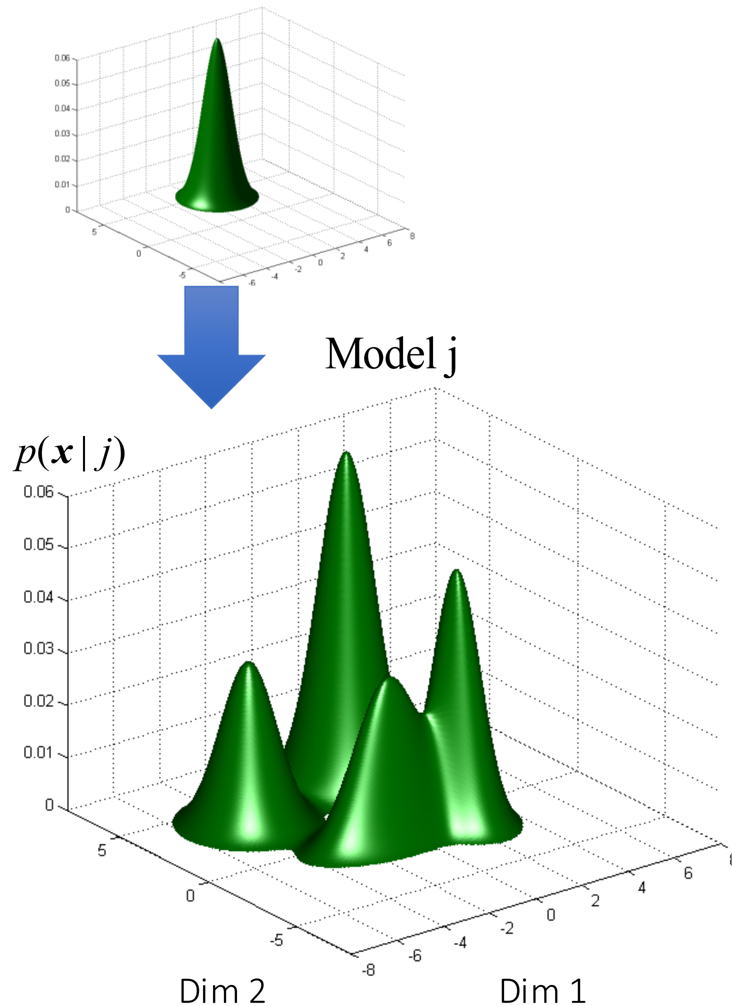


Application : GMMs for speaker recognition

- A Gaussian mixture model (GMM) represents as the weighted sum of multiple Gaussian distributions

- Each Gaussian state i has a
  - Mean μ_j
  - Covariance Σ_j
  - Weight

$$w_j \square p(\mu = \mu_j)$$



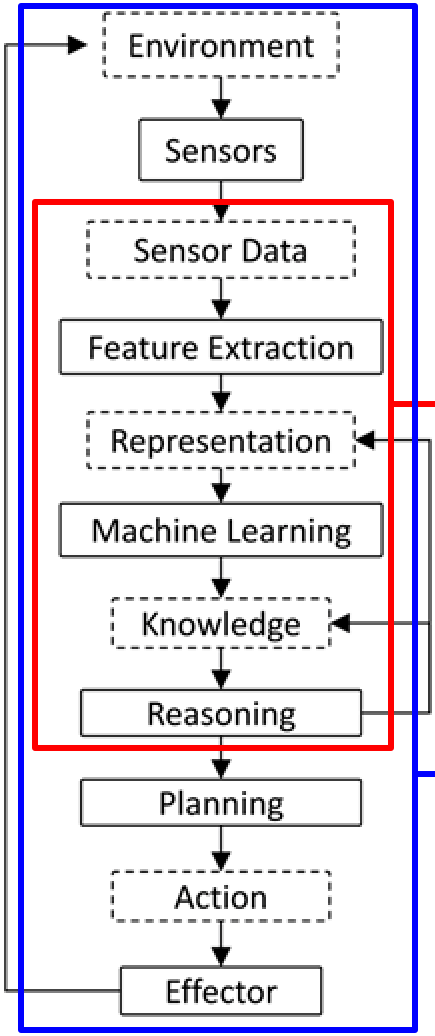
GMM Clustering – EM based optimization

Data → Task → Representation → Score Function → Search/Optimization → Models, Parameters

Tabular → Clustering → Mixture of Gaussian → Likelihood → EM algorithm → Each point's soft membership & mean / covariance per cluster

$$\sum_i \log \prod_{i=1}^{n} p(x=x_i) = \sum_i \log \left[\sum_{\mu_j} p(\mu=\mu_j) \frac{1}{(2\pi)\left|\Sigma_j\right|^{1/2}} e^{-\frac{1}{2}\left(\vec{x}-\vec{\mu}_j\right)^T \Sigma_j^{-1}\left(\vec{x}-\vec{\mu}_j\right)} \right]$$

The Full Stack



The promise of **Deep Learning**

The promise of **Deep Reinforcement Learning**

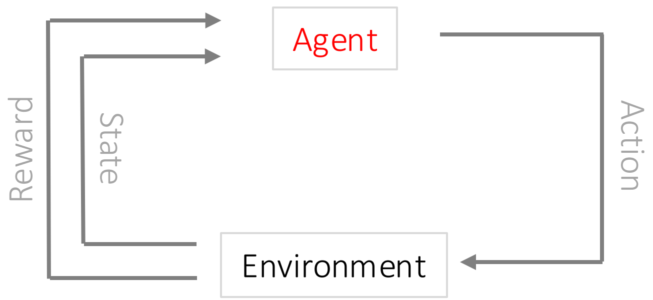
Reinforcement Learning

- Learning to interact with an environment
  - Robots, games, process control
  - With limited human training
  - Where the 'right thing' isn't obvious

- Supervised Learning:
  - Goal: $f(x) = y$
  - Data: $[< x_1, y_1 >, \dots, < x_n, y_n >]$

- Reinforcement Learning:
  - Goal:
    Maximize $\sum_{i=1}^{\infty} Reward(State_i, Action_i)$

  - Data:
    $Reward_i, State_{i+1} = Interact(State_i, Action_i)$



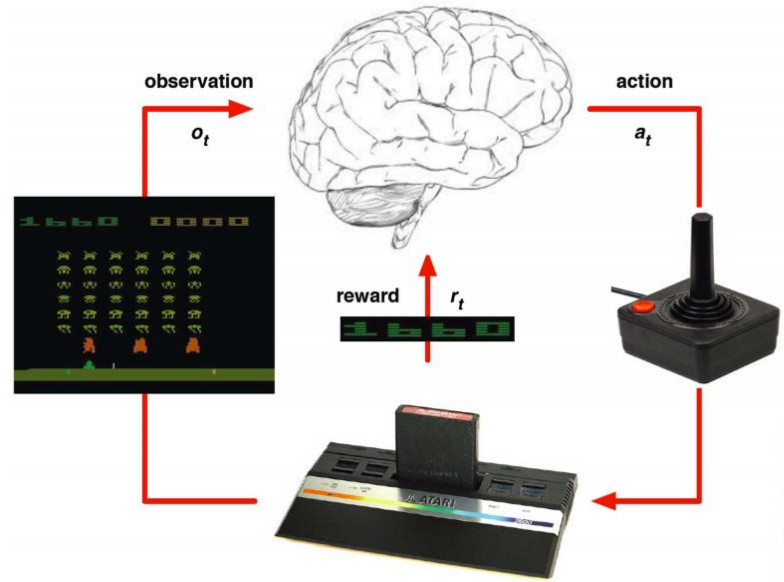
credit: Geoff Hulten

What is special about RL?

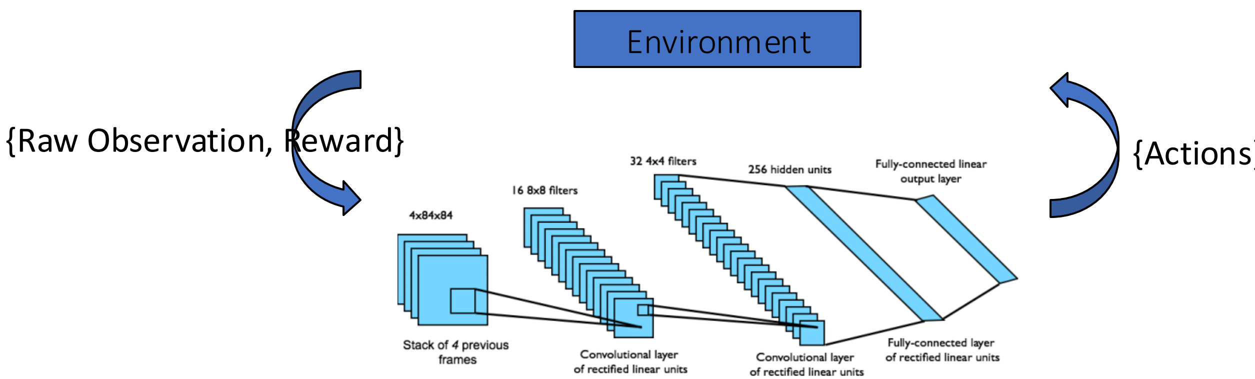
- RL is learning how to map states to actions, so as to maximize a numerical reward over time.
- Unlike other forms of learning, it is a multistage decision-making process (often Markovian).
- An RL agent learn by trial-and-error. (Not entirely supervised, but interactive)
- Actions may affect not only the immediate reward but also subsequent rewards (Delayed effect).

Deep Reinforcement Learning

- Human



- So what's **DEEP** RL?



Adapt from Professor Qiang Yang of HK UST

Elements of RL

- A policy
  - A map from state space to action space.
  - May be stochastic.
- A reward function
  - It maps each state (or, state-action pair) to a real number, called reward.
- A value function
  - Value of a state (or, state-action pair) is the total expected reward, starting from that state (or, state-action pair).

The Precise Goal / Popular RL Algorithms

- To find a policy that maximizes the Value function.
    - transitions and rewards usually not available

- There are different approaches to achieve this goal in various situations.

- Value iteration and Policy iteration are two more classic approaches to this problem. But essentially both are dynamic programming.

- Q-learning is a more recent approaches to this problem. Essentially it is a temporal-difference method.

12/02

Today's Roadmap

- 0. Logistics:
  - Dec. 9th Final Exam in Person
    - (printed / manual notes only!)
    - Similar to quiz questions, but some a bit longer
  - Please sign up for your project final presentation time slot ASAP!
  - HW5 Due tomorrow
  - We will use your top10 quiz grades … only top 10 !!

- 1. Today review:
  - a. RL + RL Gym (not part of final exam!) --- Tuesday
  - b. a quick survey on the contents
  - c. makeup quiz Q14 Dec. 2nd

- This Thursday we will go over all your QA questions + past quizzes + Q15

HW5 – minor notes

- In the unlabeled sample set "salary.2Predict.csv", last column includes a fake field for class labels. You are required to generate/ predict labels for samples in "salary.2Predict.csv". **Remember not to shuffle your prediction outputs!**

- **Include the CV classification accuracy results in** your pdf report by performing **3-fold cross validation (CV)** on the labeled set "salary.labeled.csv" (including about 38k samples) , recommend **>= three different SVM kernels** you pick.

- **Provide details** about the kernels you have tried and their performance (e.g. 3CV classification accuracy) including results on **both CV train accuracy and CV test accuracy into the writing.** (**Highly recommended**: table with each row containing kernel choice, kernel parameter, CV train accuracy and CV test accuracy). **--- a result table will be handy**

- **Feel free to use some existing libraries for easier data pre-processing.**

HW5 recommend model selection process:

```
1.  load train_set, test_set

2. new_train_set, new_valid_set = data_split(train_set)

3. valid_losses = []

4. for hyper_param in hyper_param_set:
       theta* = model_train(new_train_set, hyper_param)
       valid_loss = model_eval(new_valid_set, theta*, hyper_param)
       valid_losses.append(valid_loss)

5. best_hyper = argmin(valid_losses)

6. theta* = model_train(train_set, best_hyper)

7. reported_performace = model_eval(test_set, theta*, best_hyper)
```

12/04

Today's Roadmap

- 0. Logistics:
  - Dec. 9th Final Exam in Person / One paper
    - (printed / manual notes only!)
    - similar to quiz questions, but some a bit longer
  - Please sign up for your project final presentation time slot ASAP!

- 1. Today review:
  - A. go over all your QA questions + past quizzes + Q15