UVA CS 4774: Machine Learning

Part 2: Course Session QA

Related:

- Part 1 [HERE]
- Course Announcement Page [HERE]

Dr. Yanjun Qi

University of Virginia

Department Of Computer Science

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

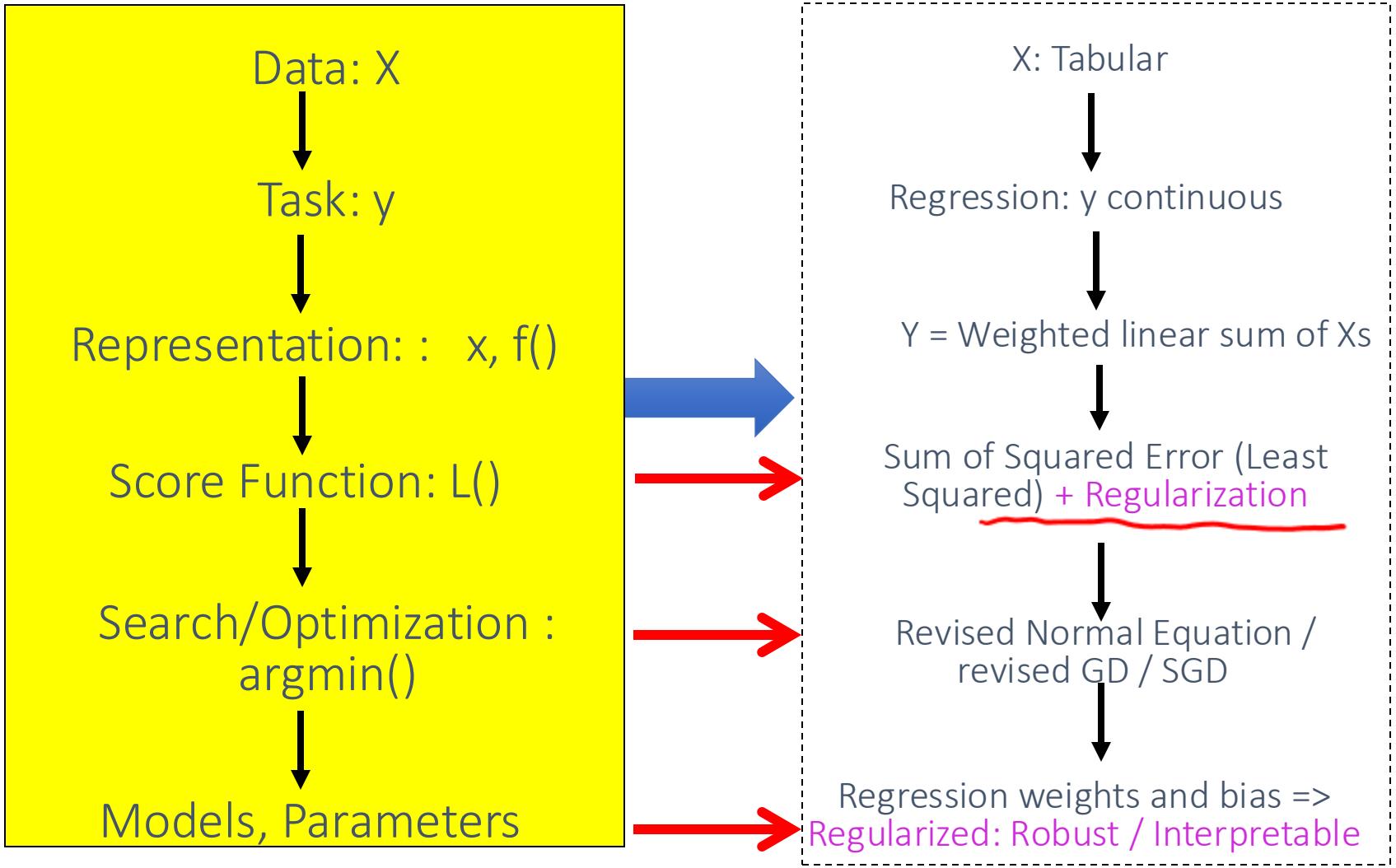
- Format:
  - Team, individual …
  - Shark Tank alike Screening ? – https://en.wikipedia.org/wiki/Shark_Tank
  - Next week – Idea collection

- 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



L7: Regularized multivariate linear regression

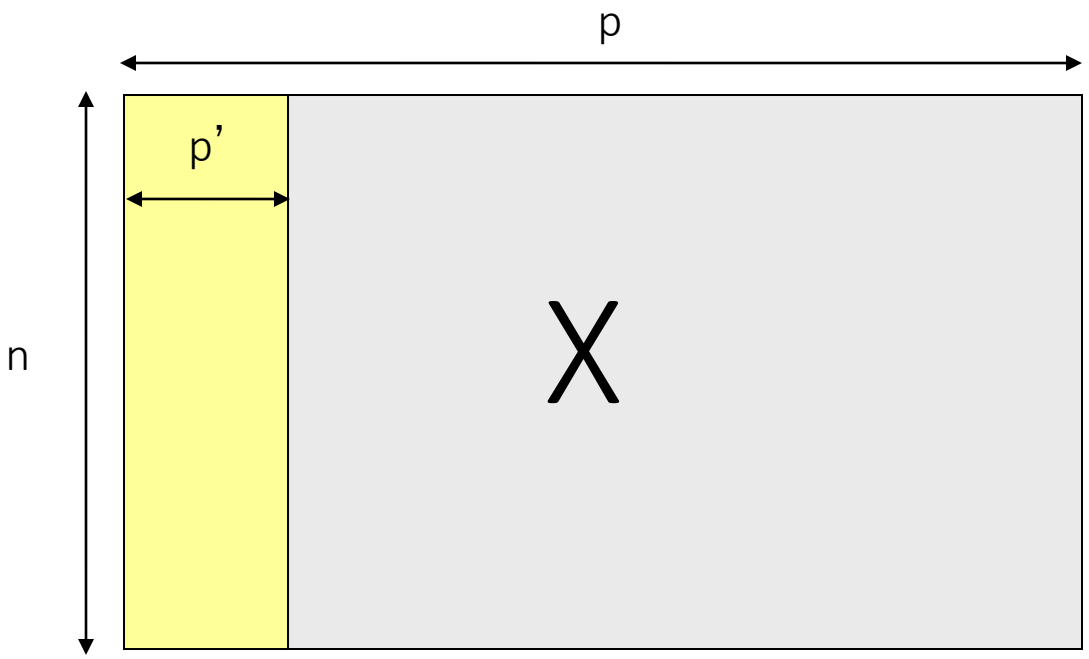
We aim to make our trained model

- 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?



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: $\arg\min \sum_{i=1}^{n} \left(Y - \hat{Y} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$
  - Ridge regression estimation: $\arg\min \sum_{i=1}^{n} \left(Y - \hat{Y} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$

9/54

Error on data + Regularization

Ridge Regression / L2 Regularized Regression

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

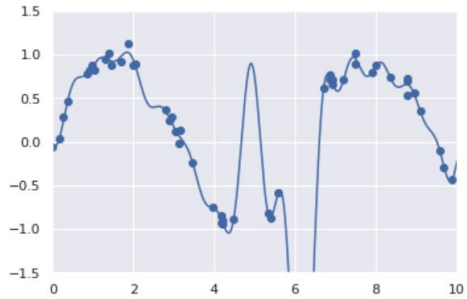
- If not invertible, a classical solution is to add a small positive element to diagonal

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

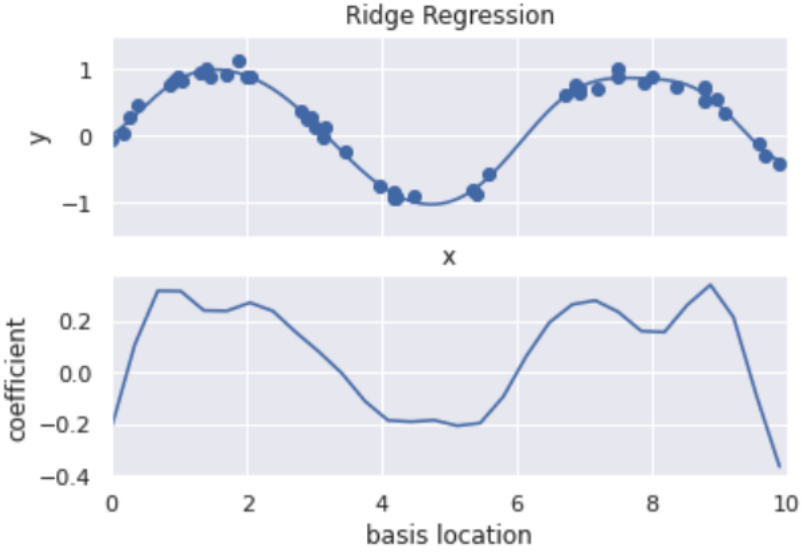
By convention, the bias/intercept term is typically not regularized. Here we assume data has been centered … therefore no bias term

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




```
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

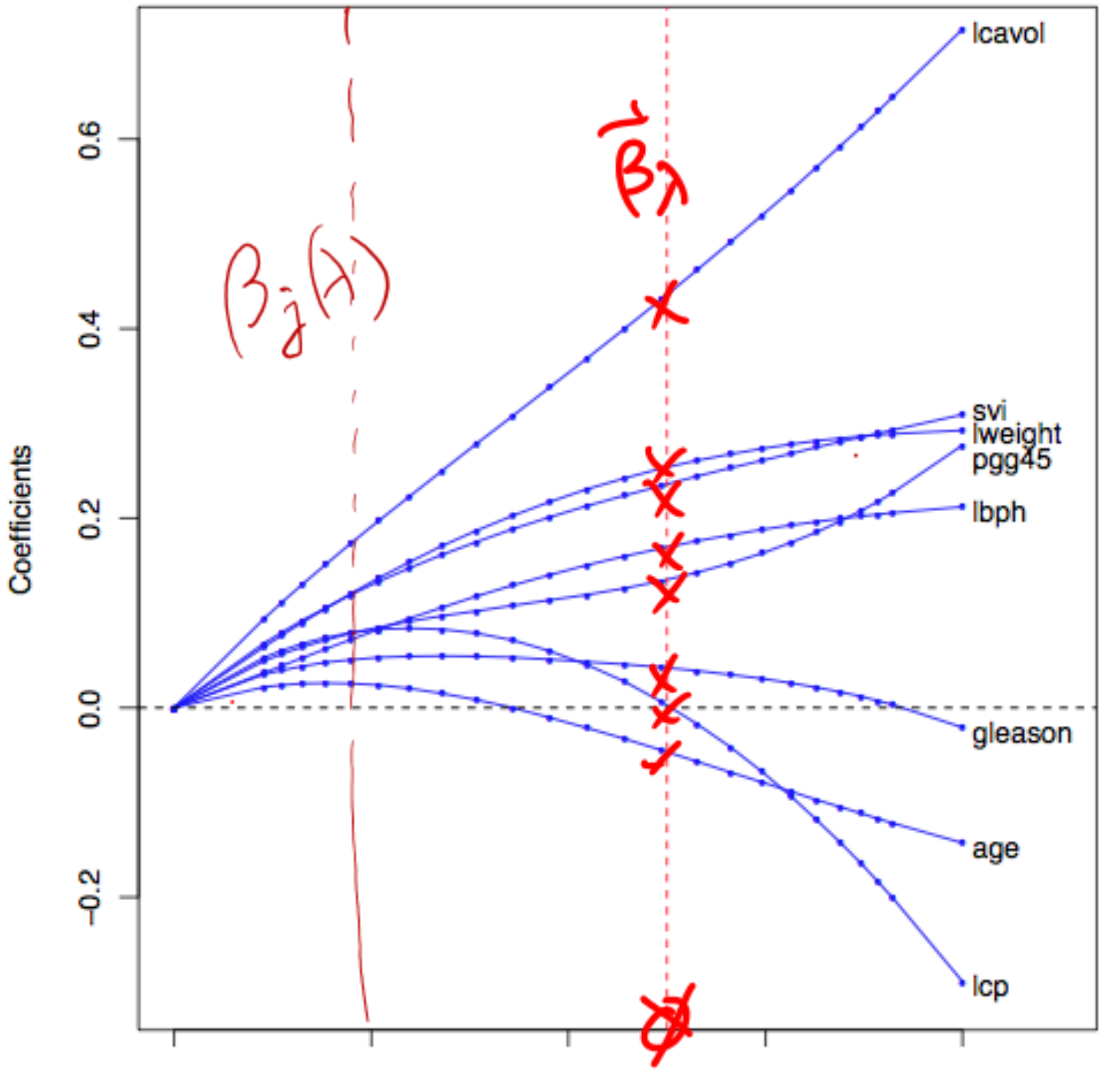
$$\square \ \beta_j^2$$
$$j$$

$$\square \ |\beta_j|$$
$$j$$

Regularization path of a Ridge Regression

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

Weight Decay



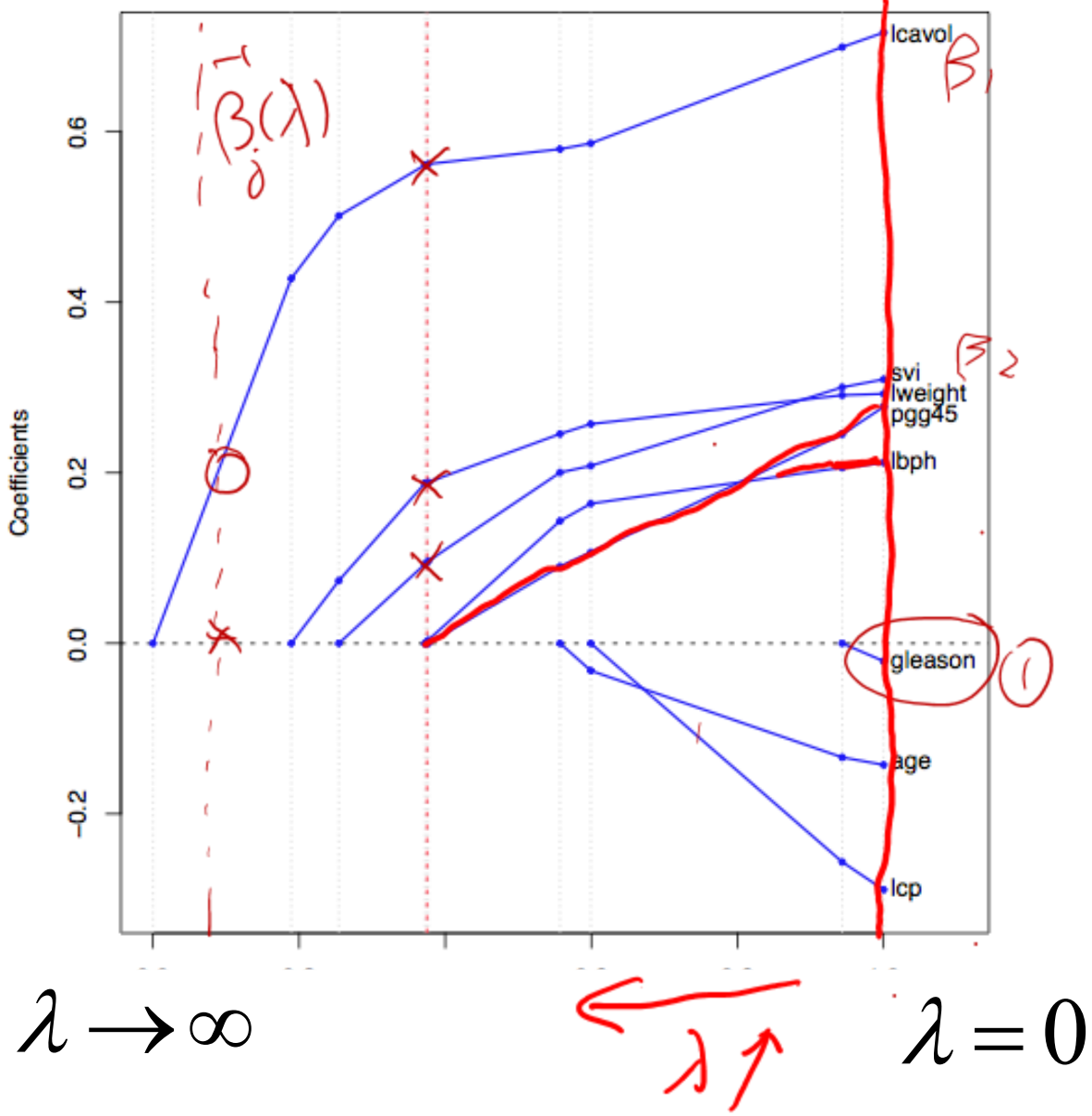
WHY and How to Select λ?

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

An example with 8 features

Regularization path of a Lasso Regression

when varying λ, how β_j varies.

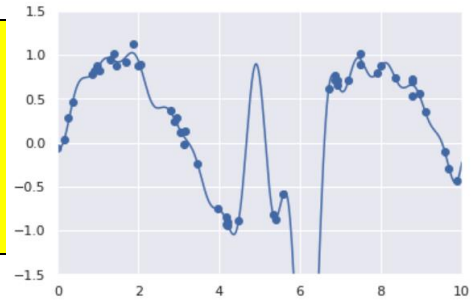


$p = 8$

An example with 8 features

Overfitting: Can be Handled by Regularization

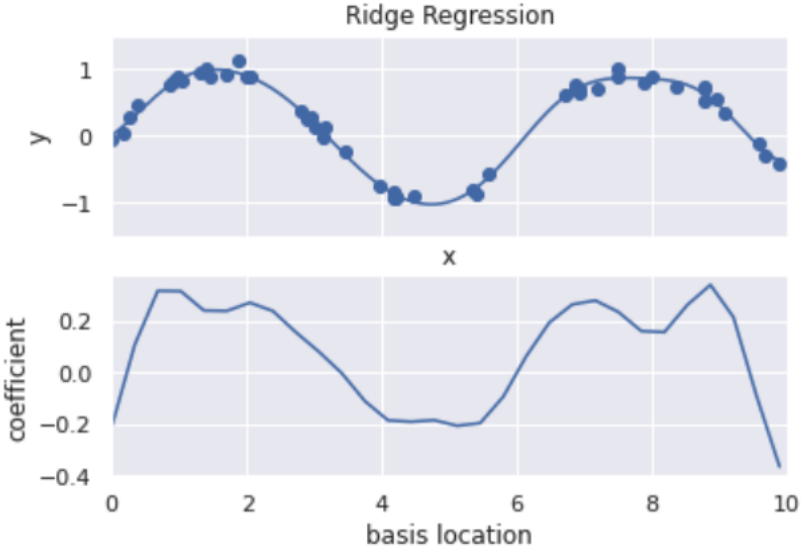
A regularizer is an additional criteria to the loss function to make sure that we don't overfit. It's called a regularizer since it tries to keep the parameters more normal/regular



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

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

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

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

L1 norm

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

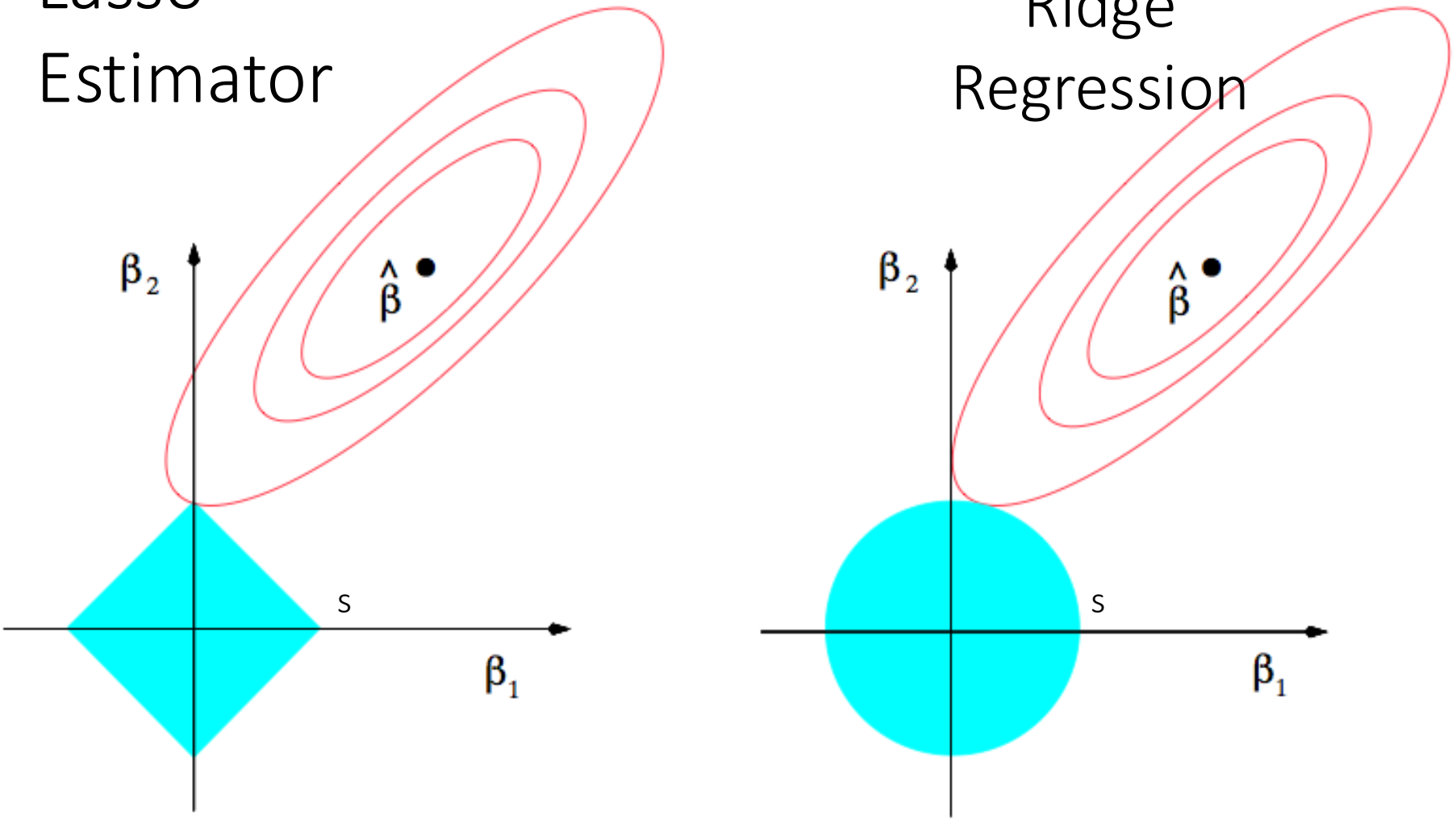
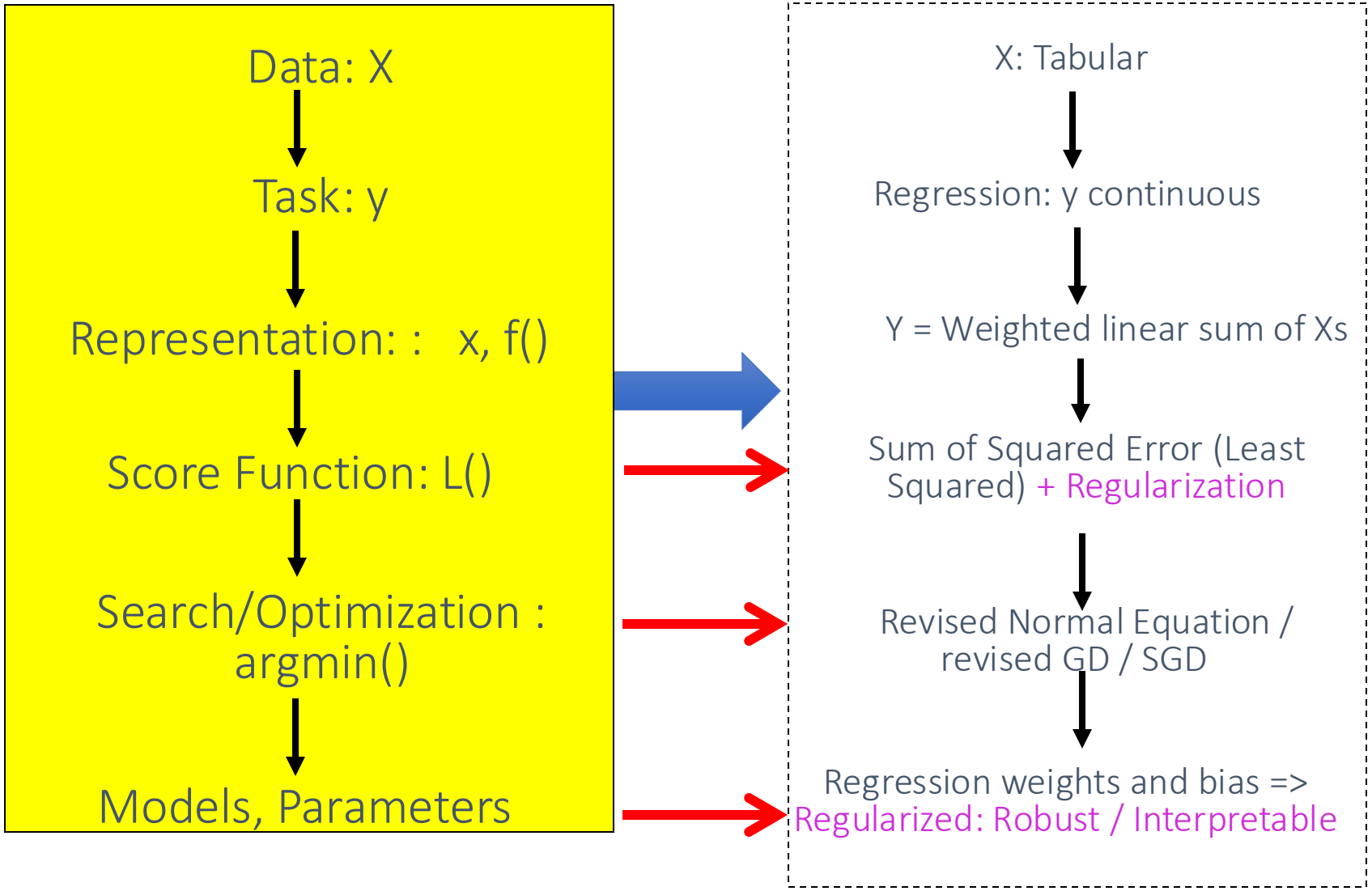


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are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



More: A family of shrinkage estimators

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

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

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

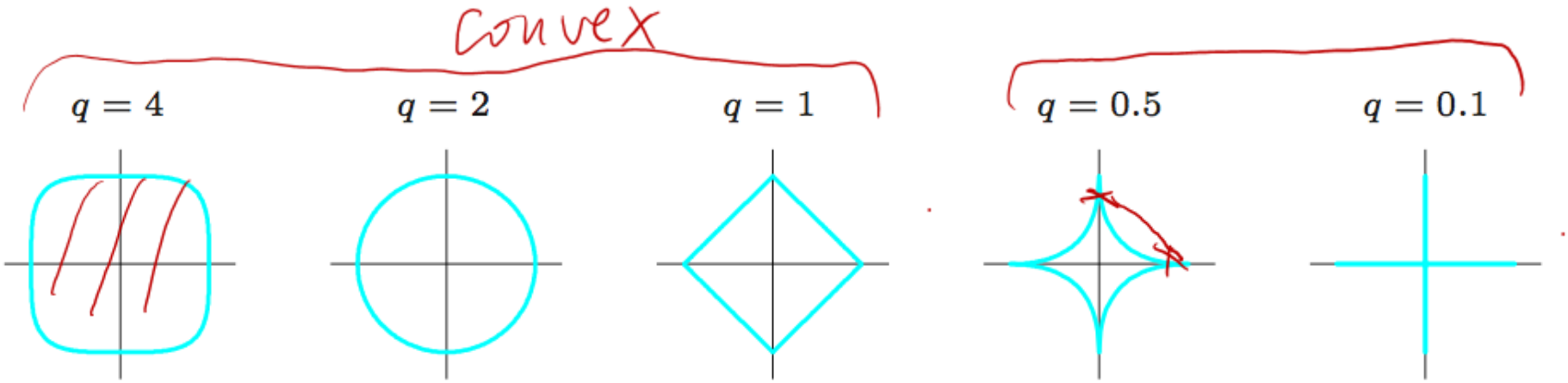
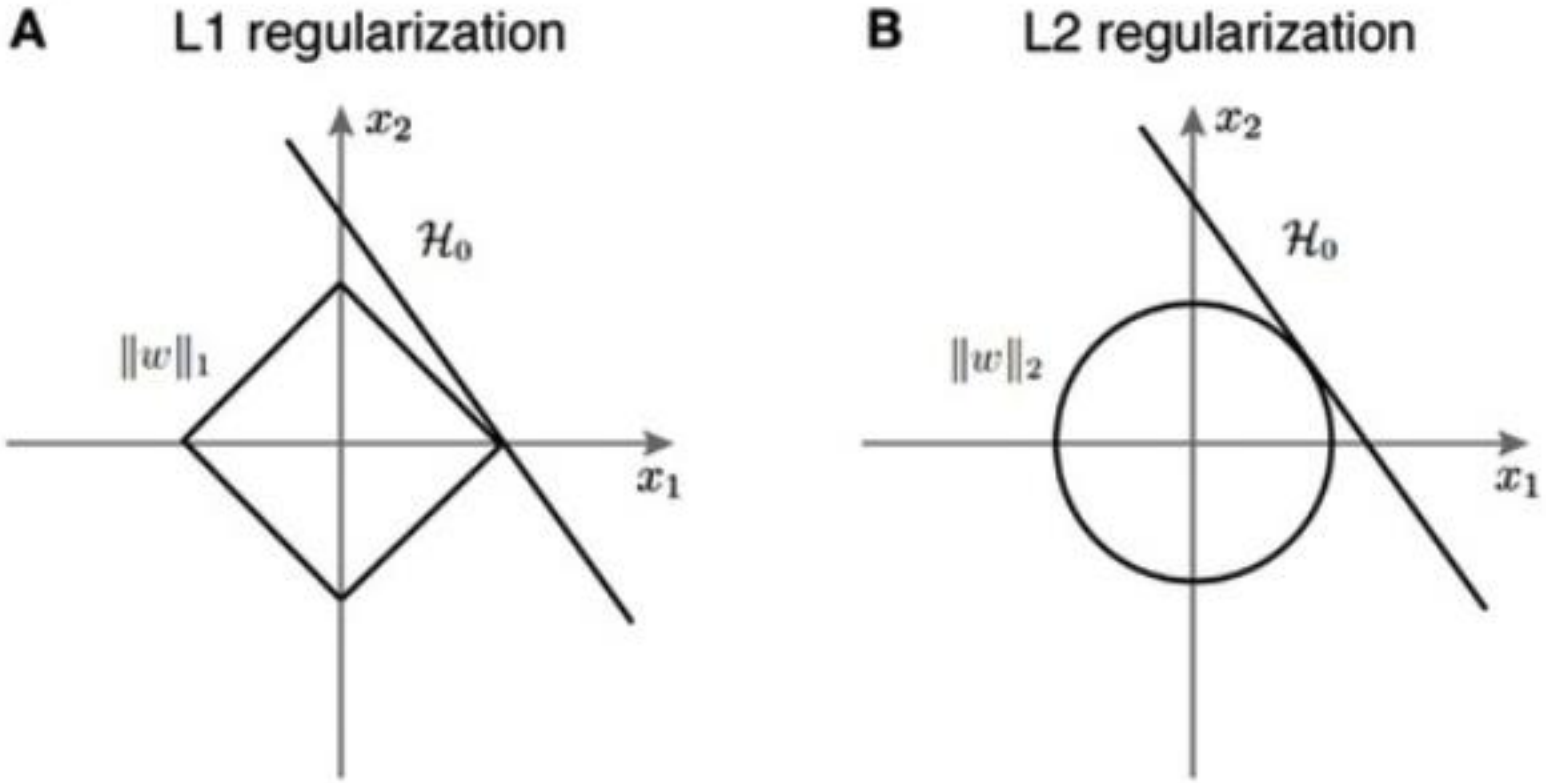


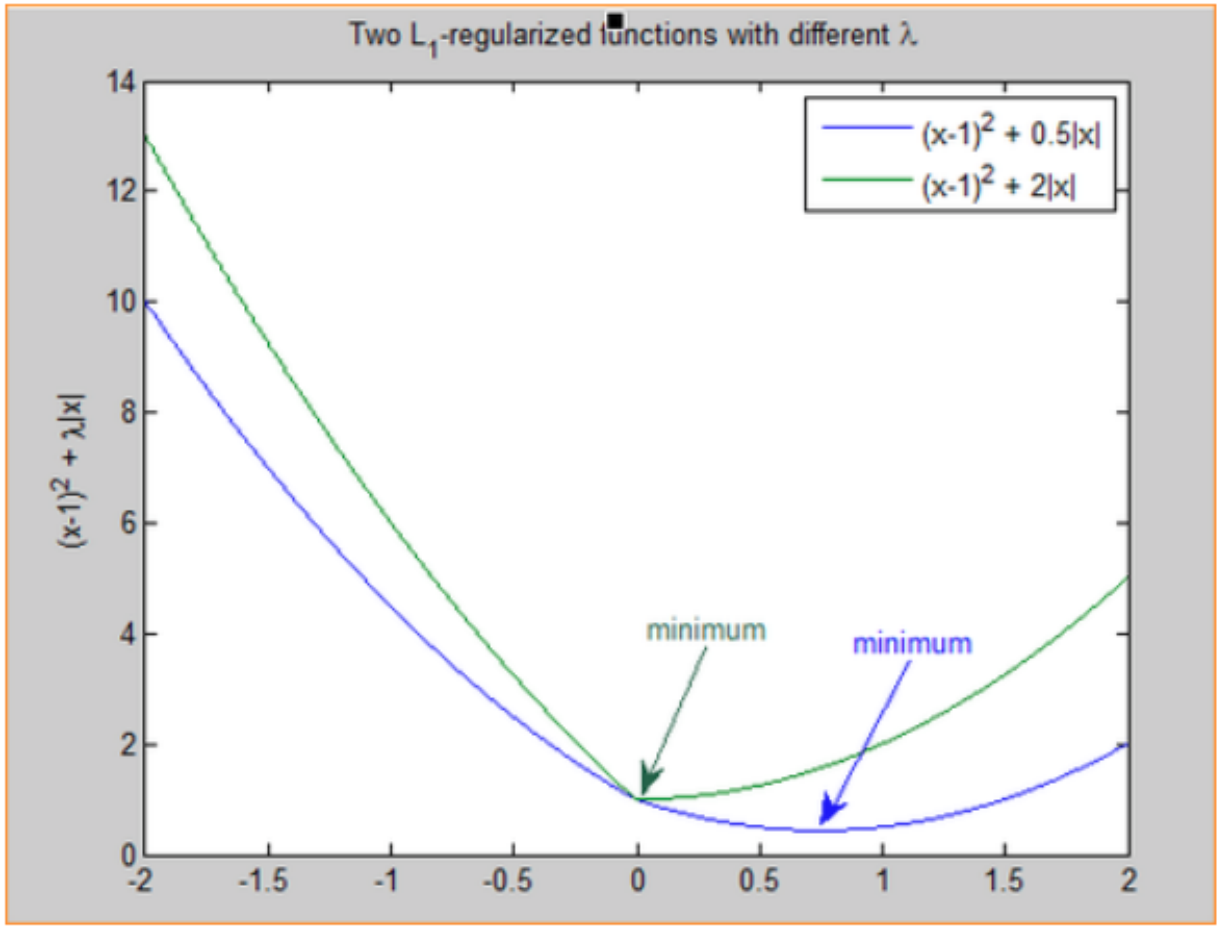
FIGURE 3.12. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for given values of* q.



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, $c_j > \lambda$

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

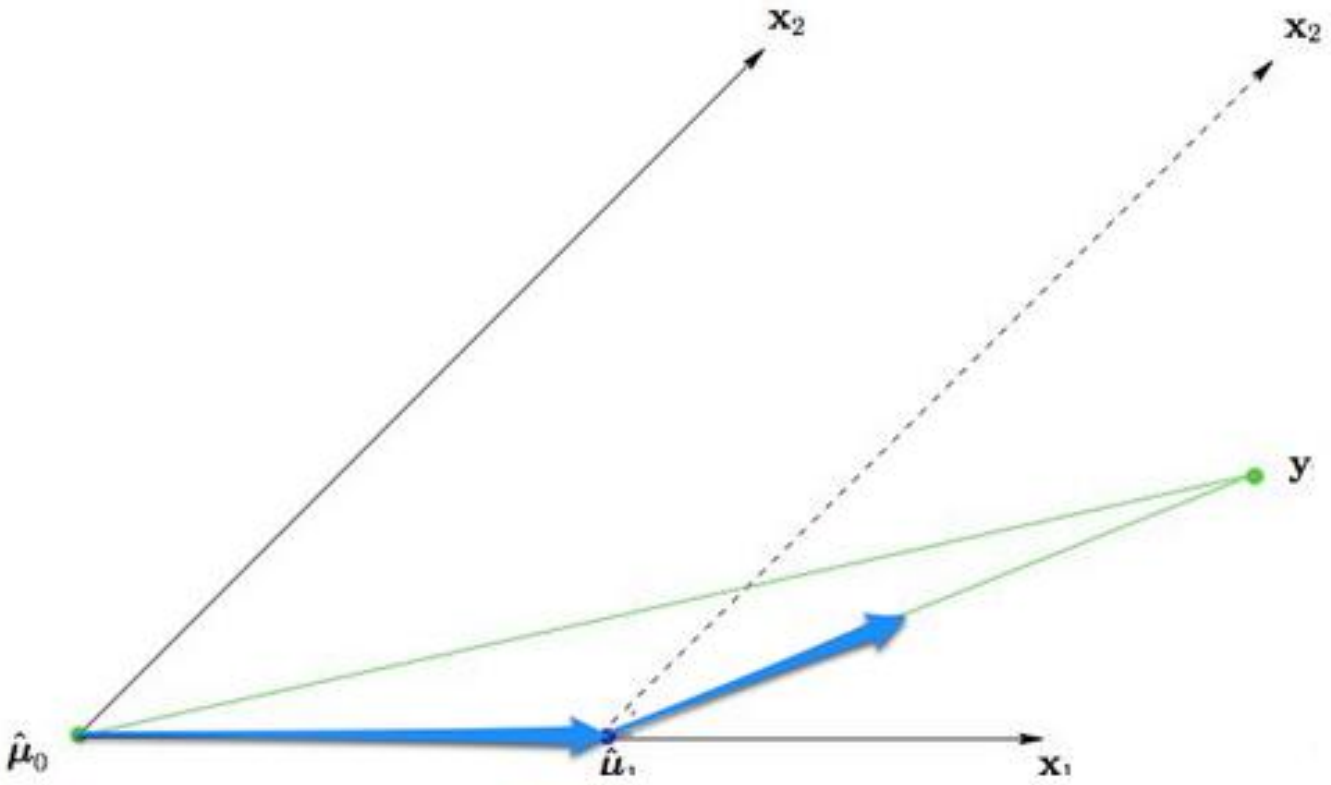
else, soft-thresholding

$$\beta_j = 0$$

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

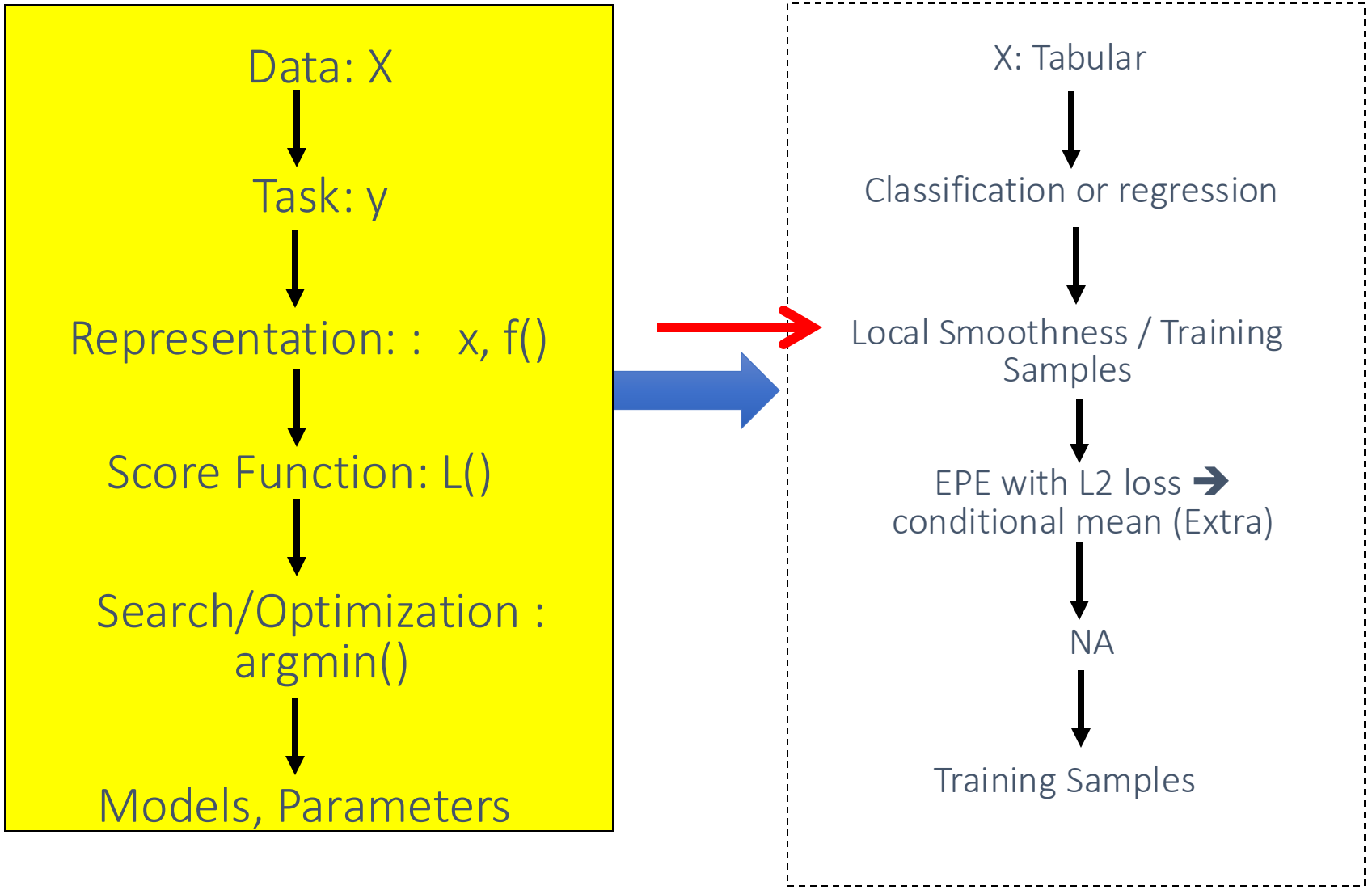
One uses different coordinate directions cyclically throughout the procedure.

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)

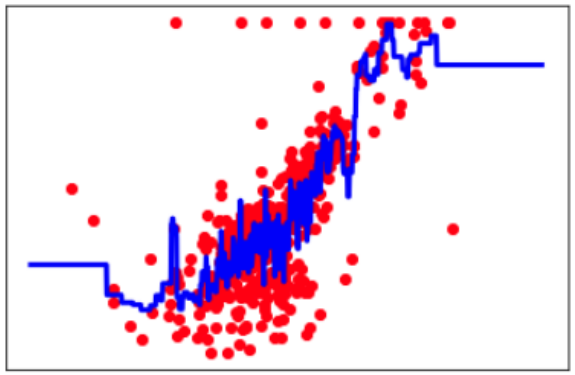


Code run: https://github.com/qiyanjun/2025Fall-UVA-CS-MachineLearningDeep/blob/main/notebook/L8_Knearest.ipynb

```
[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')
```

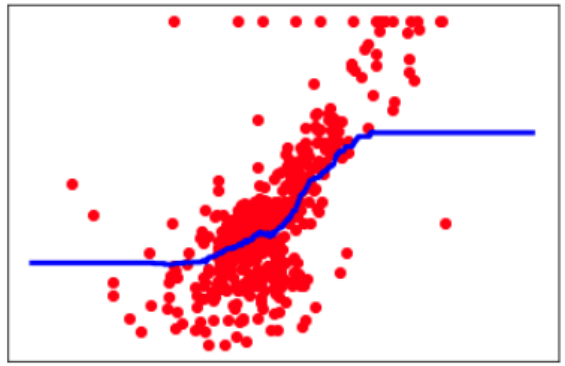
```
###
Xfit = np.linspace(3, 10, 1000).reshape(-1, 1)
yfit =knn.predict(Xfit)

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

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

plt.show()
```



```
[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

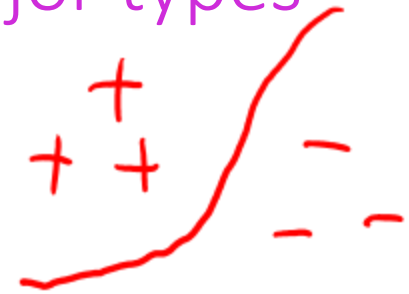
Training Mode:

- (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:

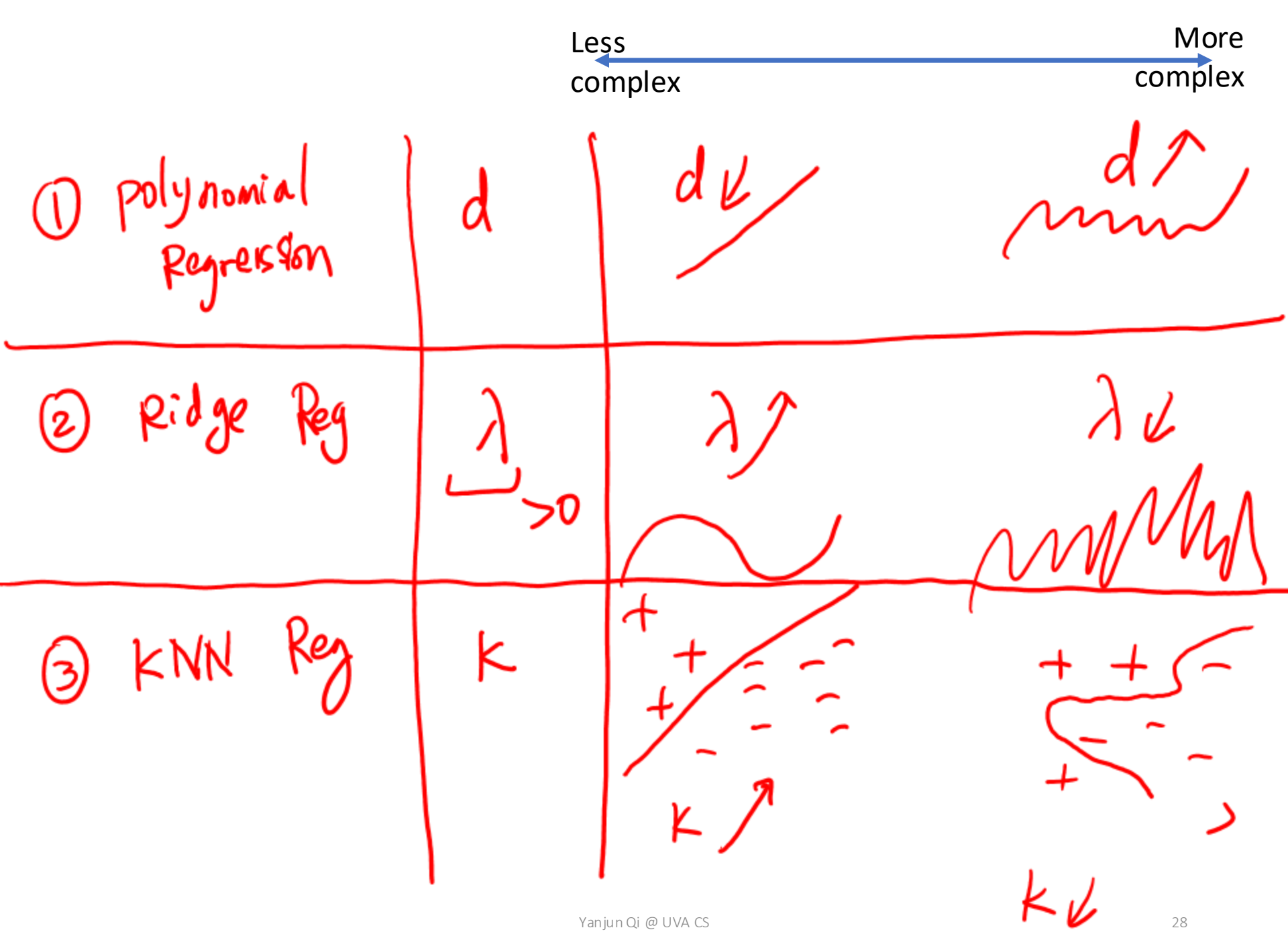
   build a generative statistical model

   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
More complex
① polynomial Regression
d
d↙
d↗
② Ridge Reg
λ
>0
λ↗
λ↙
③ KNN Reg
k
k↗
k↙

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
  - A large neighborhood → large bias → inaccurate estimation

overfit

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!

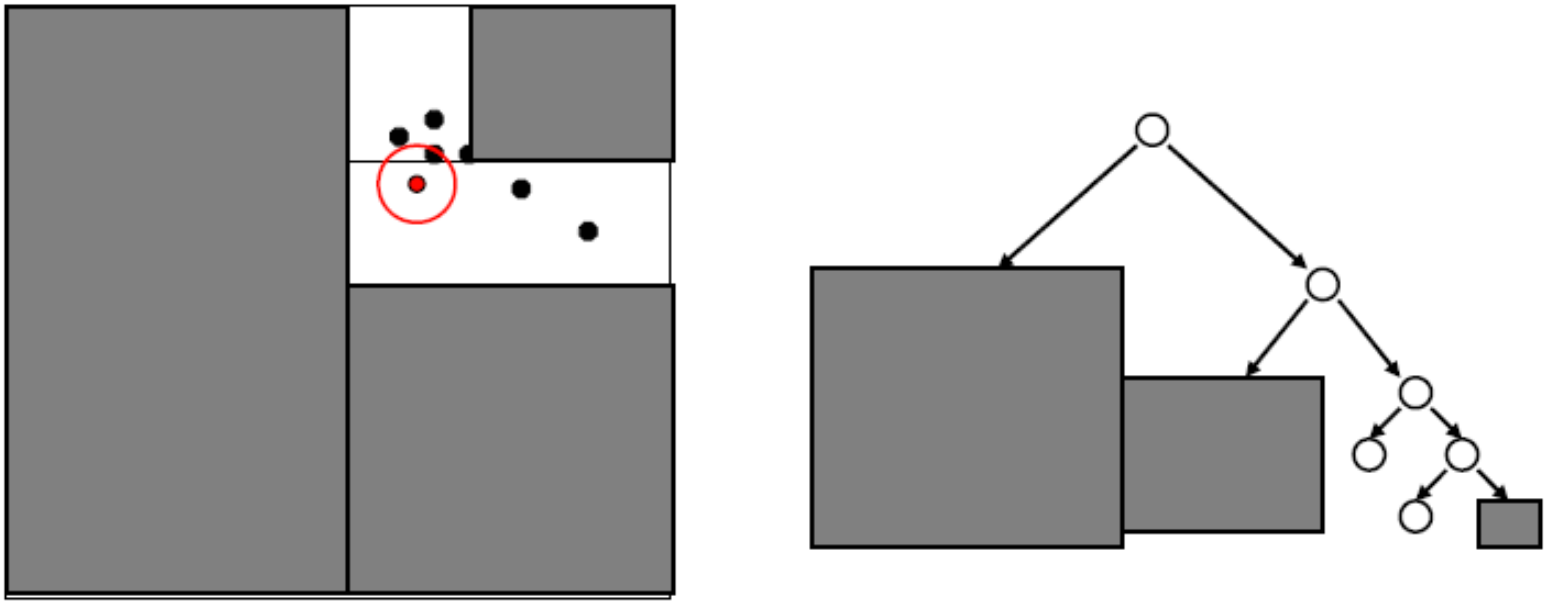
Computational Time Cost

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

$p = 30,000$

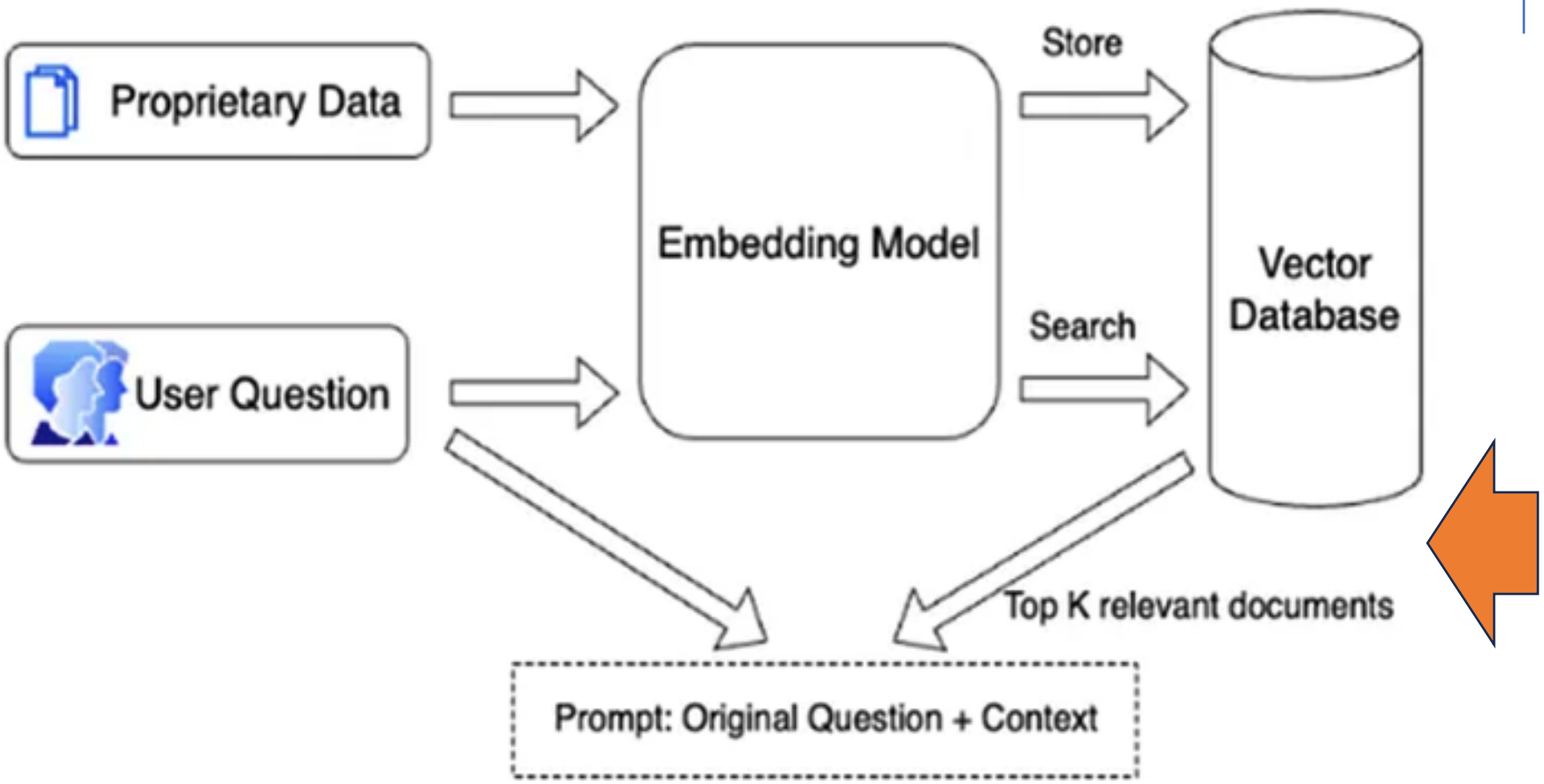
$n = 20,000$

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
- 2. Marginal probability
  - Use the total law of probability
- 3. Conditional probability
  - Use the Bayes Rule

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

$$P(B) = P(B, A) + P(B, \sim A)$$

$$= P(B, A \cup \sim A)$$

$$P(A|B)$$

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

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?

$$P(B_1 = r, B_2 = r) = P(B_1 = r)\, P(B_2 = r \mid B_1 = r) = \frac{1}{2}$$

$$\frac{3}{4} \times \frac{2}{3}$$

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

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

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.

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

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

$$\log(L(p)$$

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

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

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

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

Observed data ➔ x heads-up from n trials

Deriving the Maximum Likelihood Estimate for Bernoulli

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

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

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

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

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

$$0 = -x + pn$$

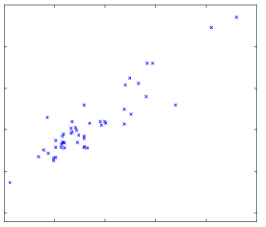
Minimize the negative log-likelihood

➔ 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:

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

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

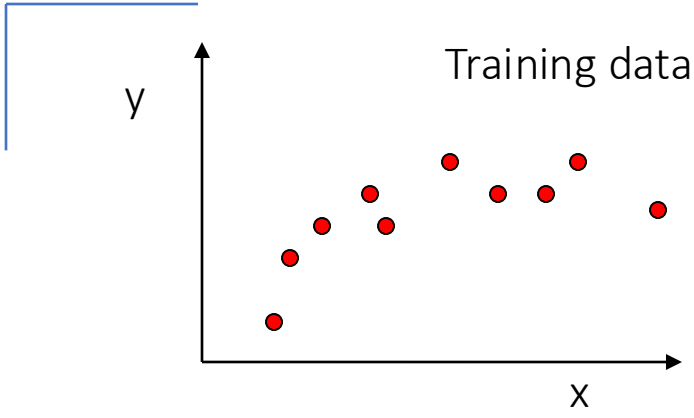
where ε is an error term of unmodeled effects or random noise

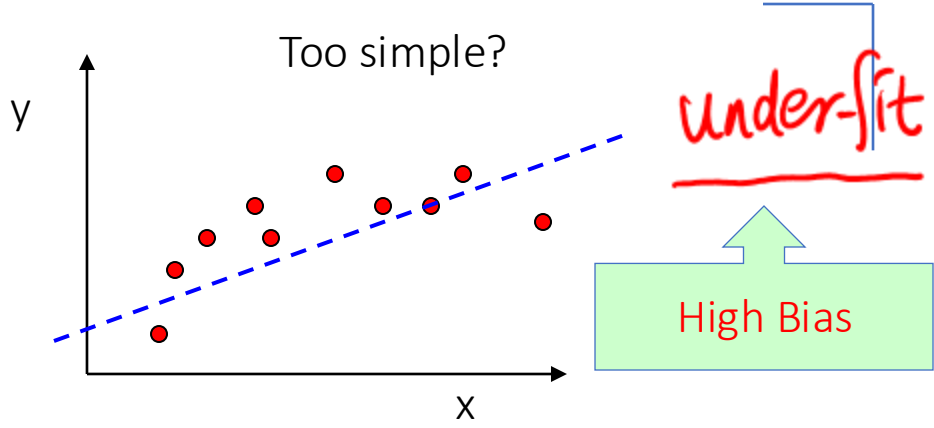
- Now assume that ε follows a Gaussian N(0,σ), then we have:

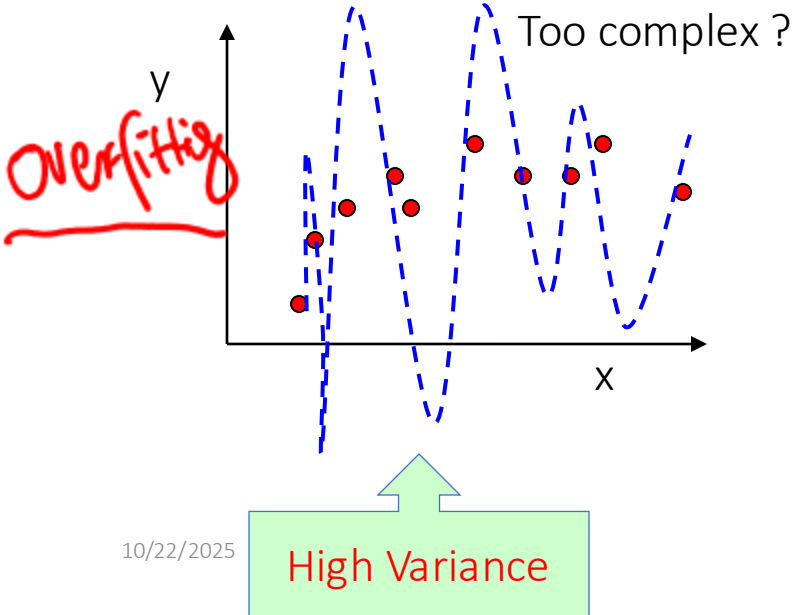
$$p(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2} \right)$$

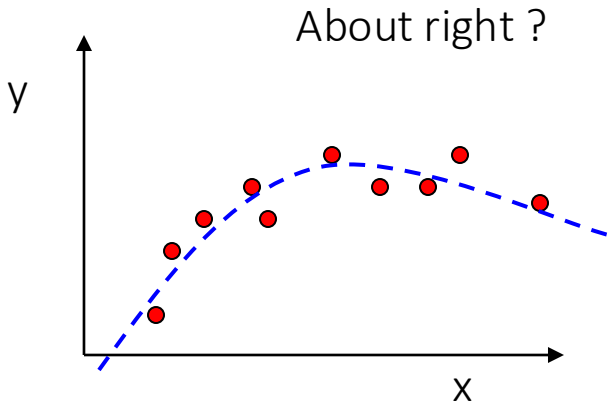
RV $y|x;\theta \sim N(\theta^T x, \sigma)$

L9: Complexity / Goodness of Fit / Generalization





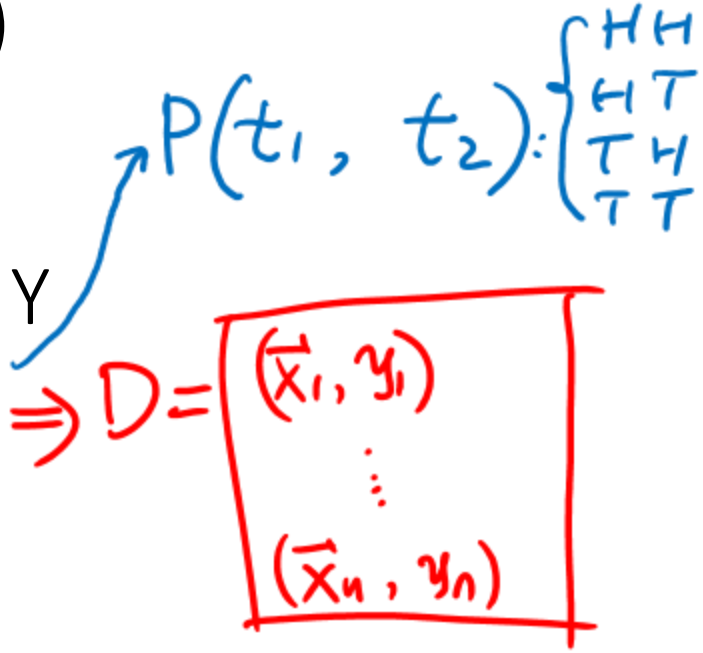




What ultimately matters: GENERALIZATION

Statistical Decision Theory (Extra)

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



- Expected prediction error (EPE):

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

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

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 → $\hat{f}$

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

Bayes Error

Irreducible / Bayes error

Bias-Variance Trade-off for EPE:

$$\text{EPE}(x) = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

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

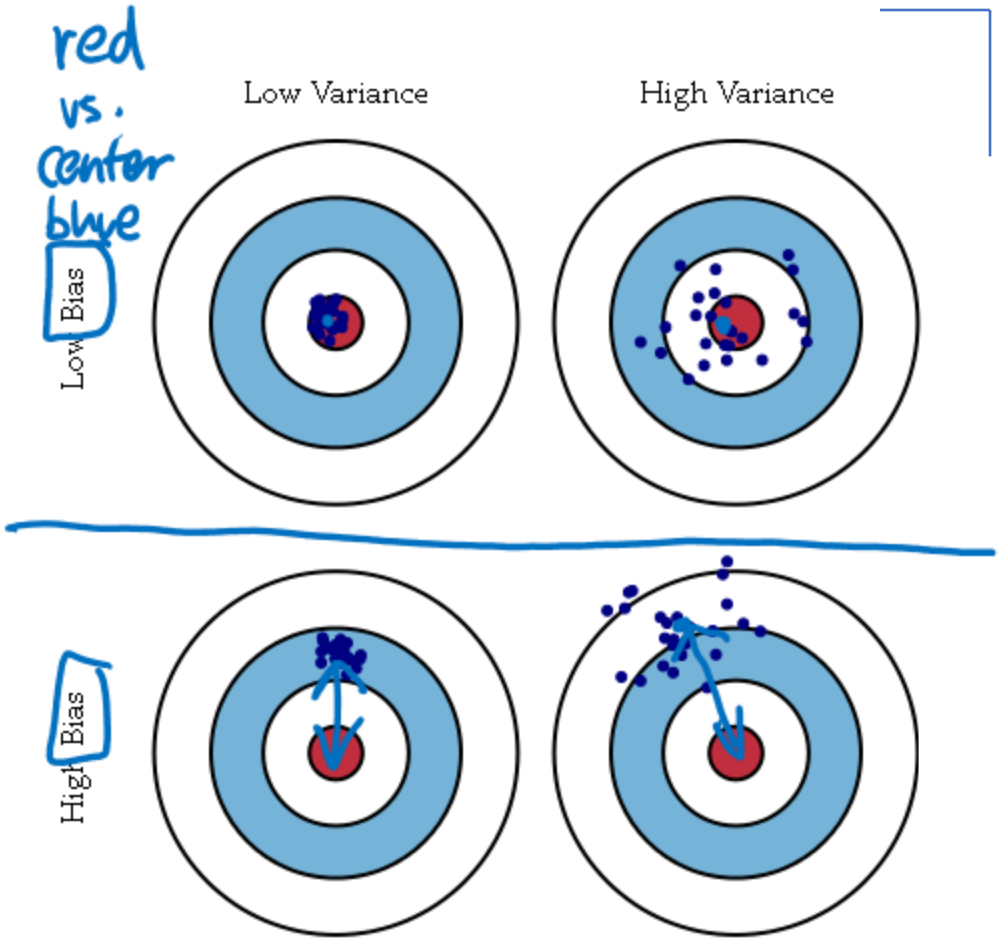
Model “bias” & Model “variance”

- 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

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



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

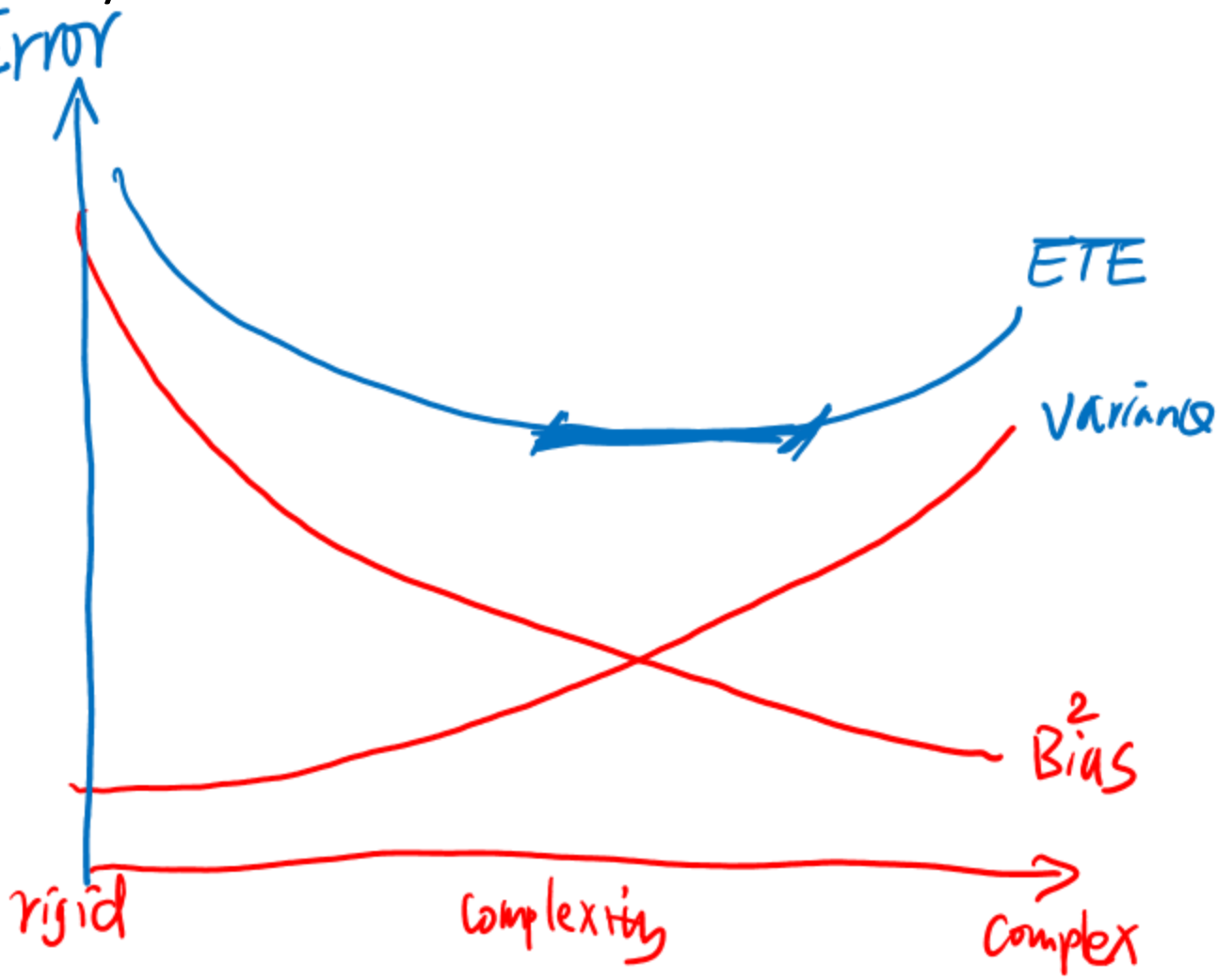
Slide credit: L. Lazebnik

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

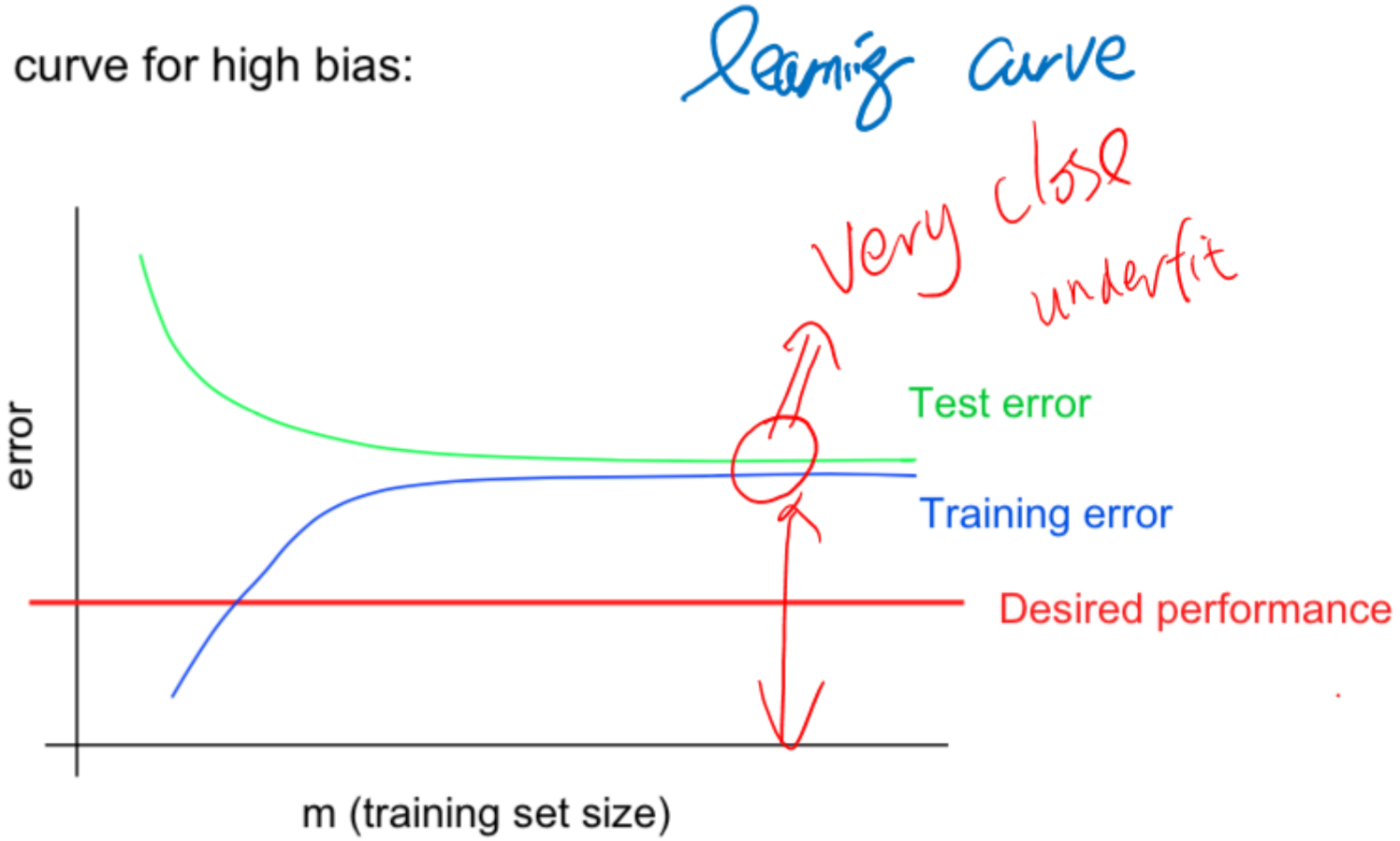
Review:
One important Control of Bias Variance Tradeoff
➔ Model Complexity

- 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:



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

High training error and high test error

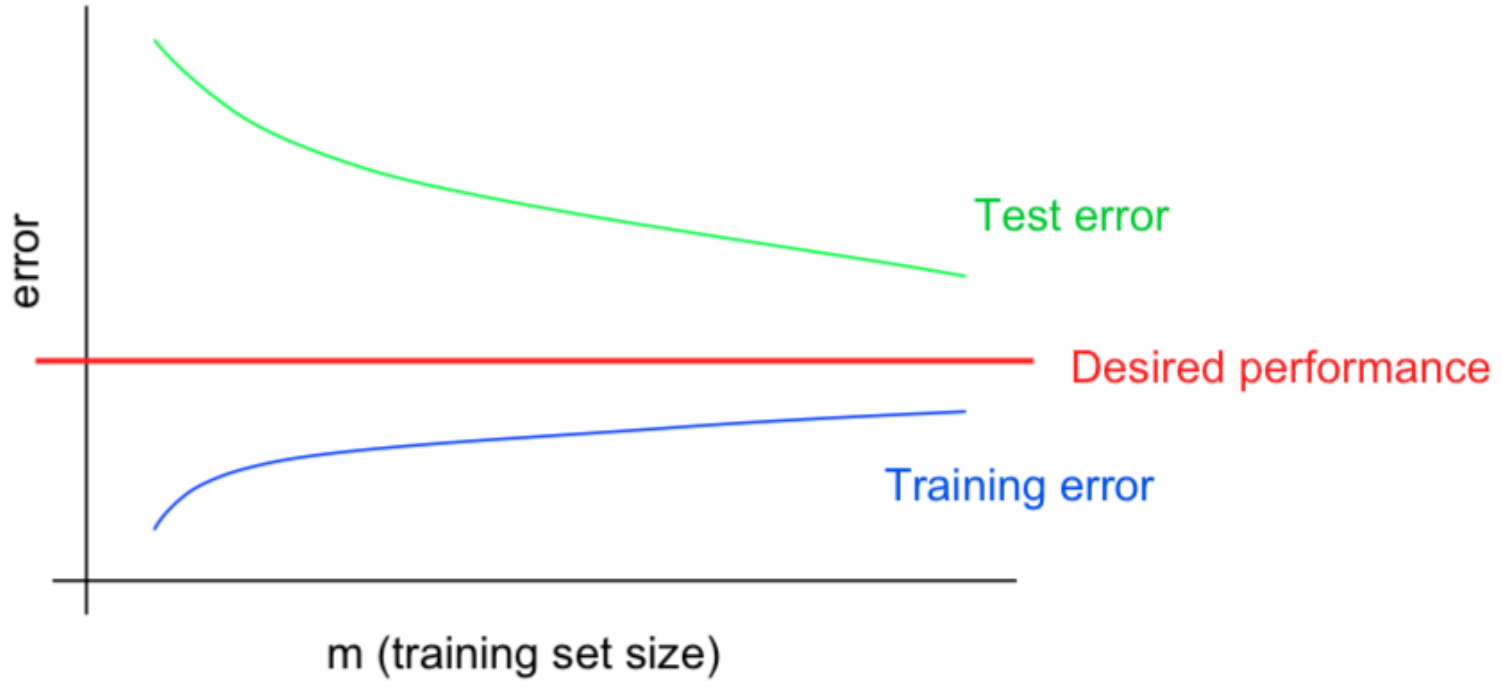
Slide credit: A. Ng

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

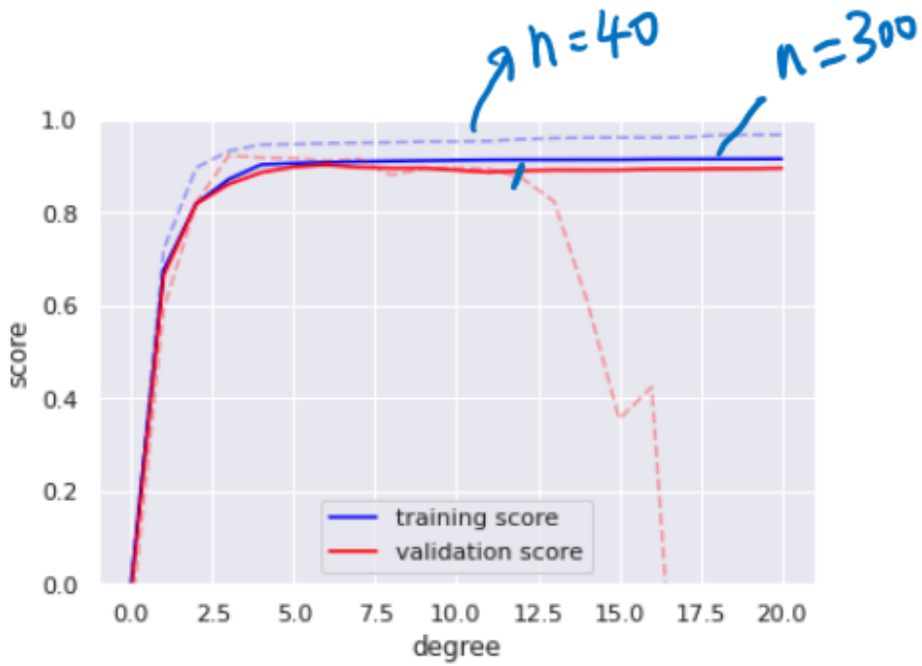
Slide credit: D. Hoiem

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

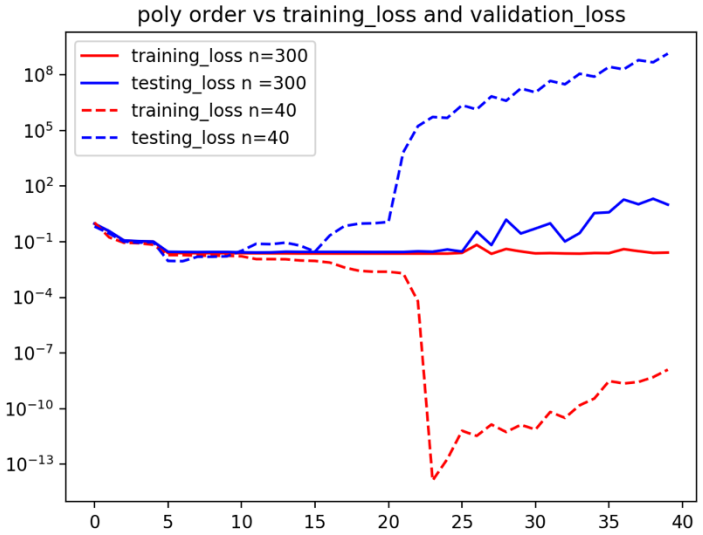
I will Code run: https://github.com/qiyanjun/2025Fall-UVA-CS-MachineLearningDeep/blob/main/notebook/L9-LearningCurves.ipynb



(1) Validation_curve_

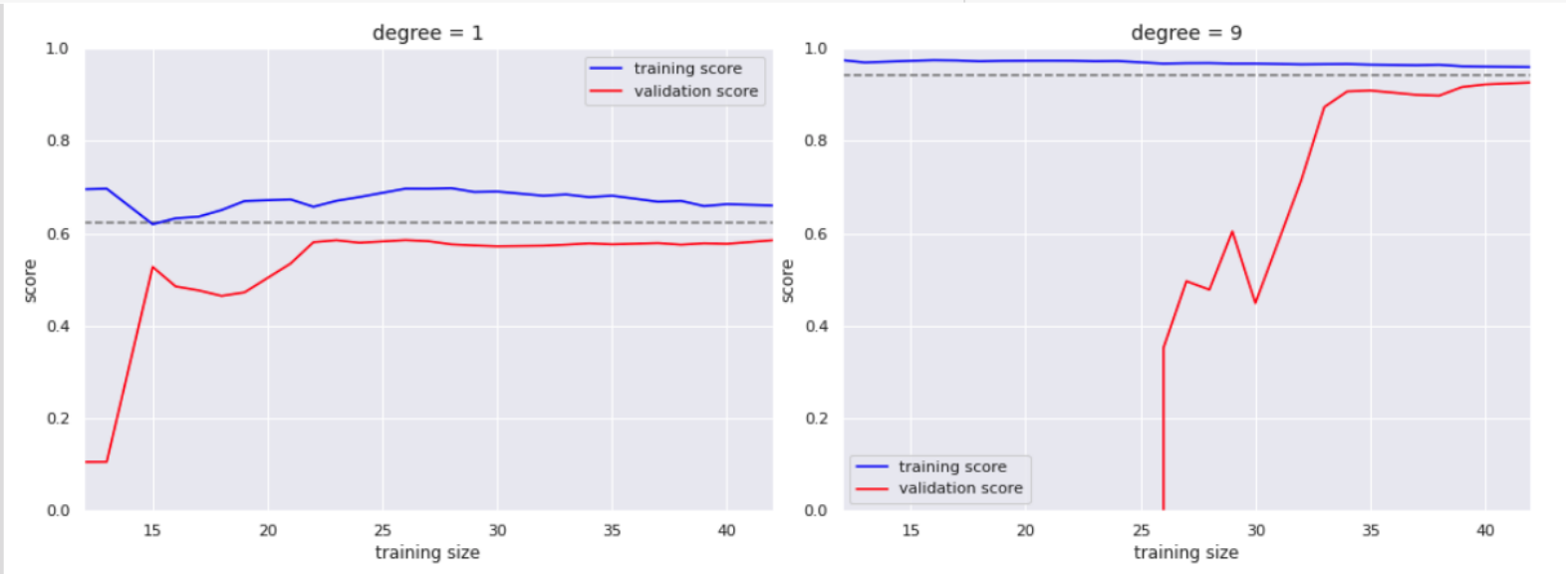
By scikitlearn Validation_curve function (normalize all metrics to positive range

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

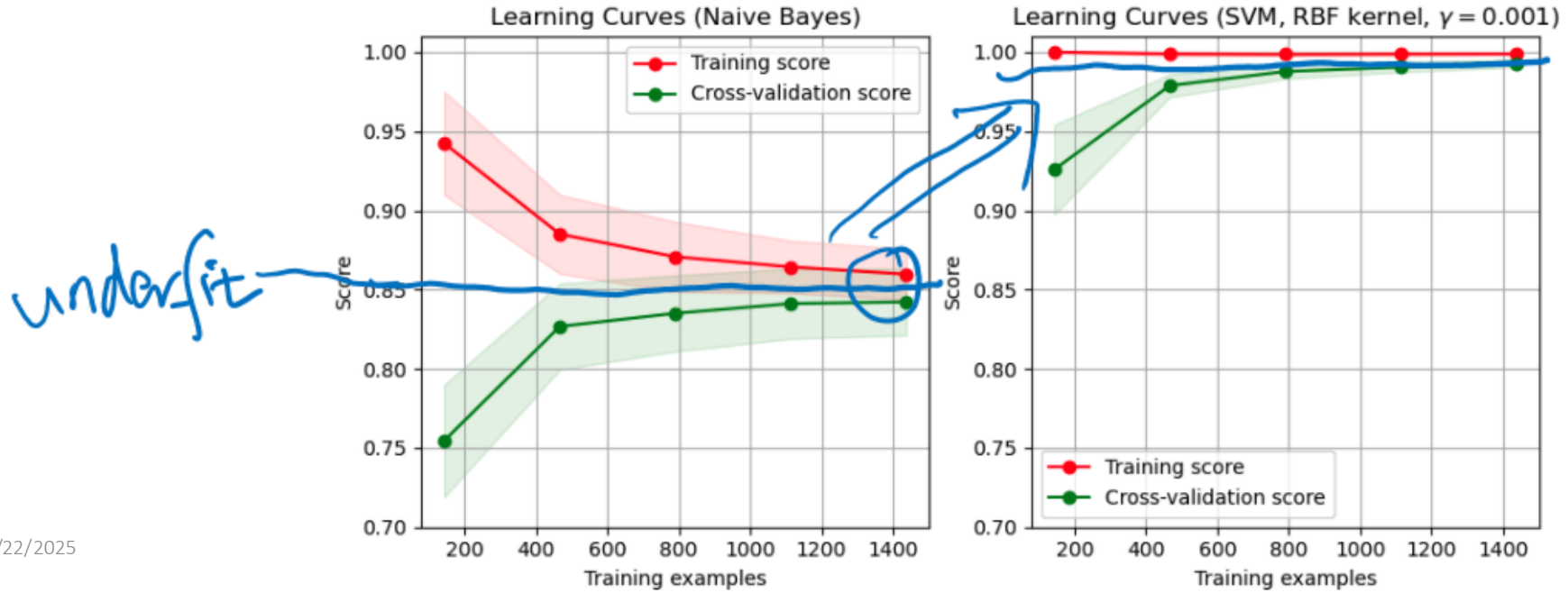


(1) Validation_curve_

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

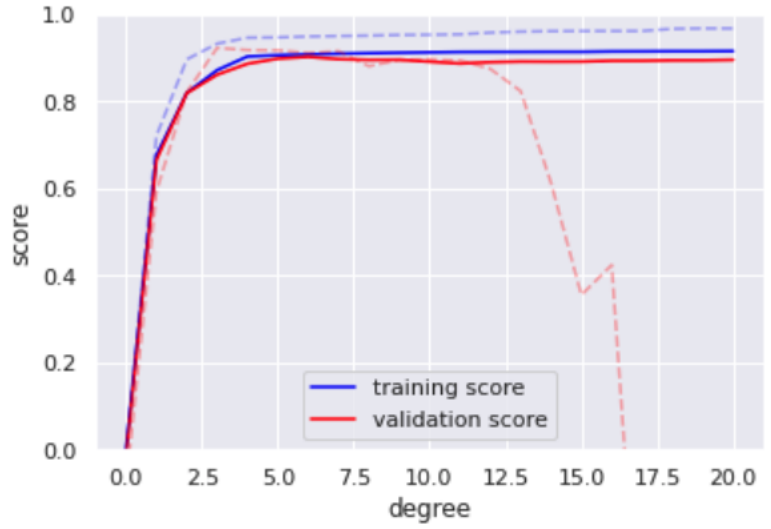


[(1) Learning Curves for polynomial regression (up) and classification (down) / by scikitlearn]



```
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');
```

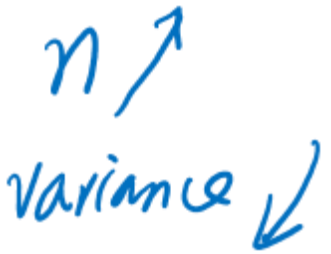

```
X2, y2 = make_data(200)
degree = np.arange(21)
train_score2, val_score2 = validation_curve(PolynomialReg
                                            'polynomialfe

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)



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

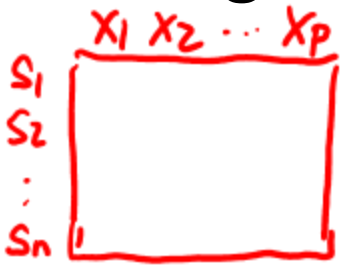
Course Content Plan ➔ Regarding Tasks

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

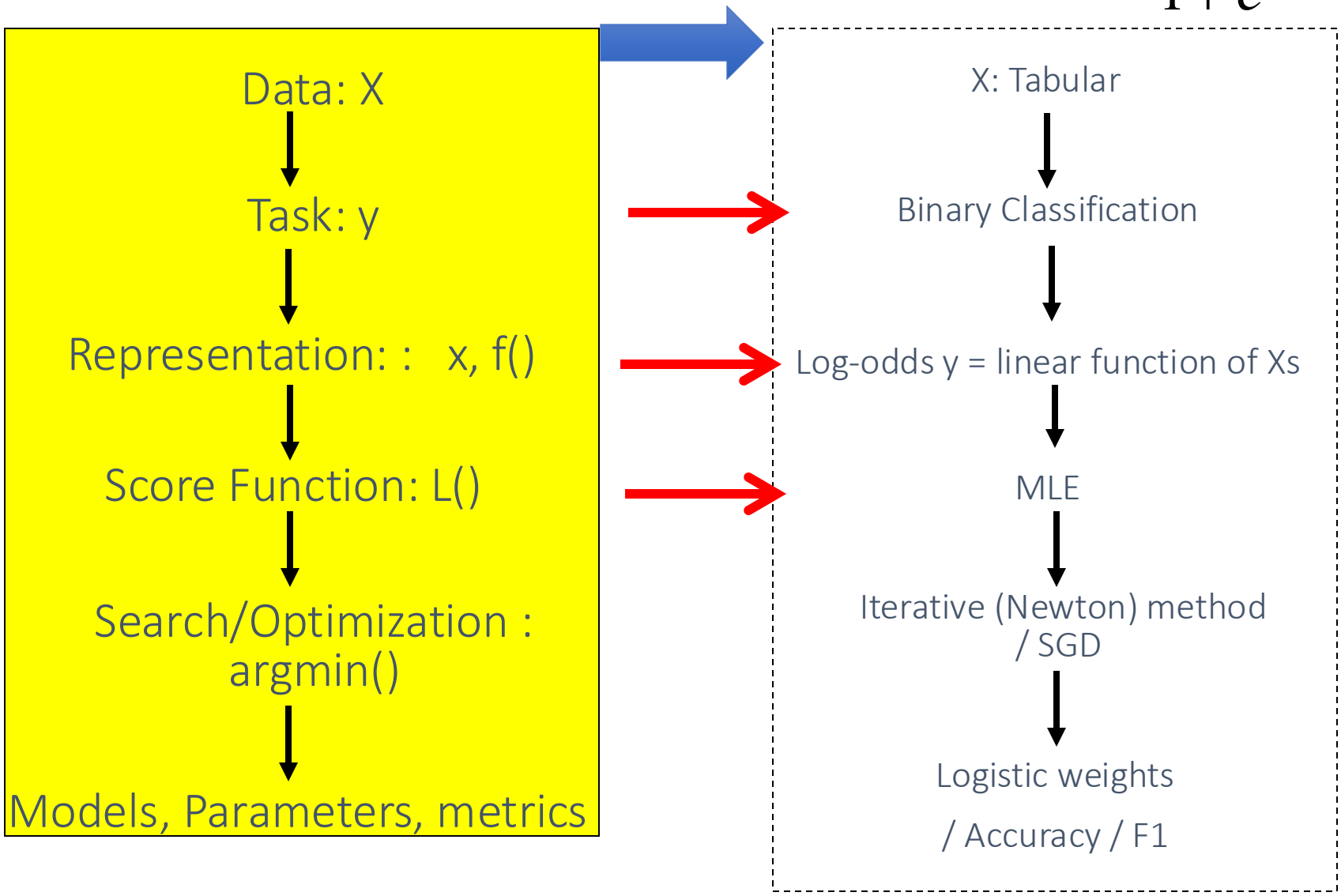


- ❑ 2D Grid Structured: Imaging

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

Today: Logistic Regression Classifier

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



Bayes Classifiers – Predict via MAP Rule

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

based on:

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

MAP Rule

MAP = Maximum Aposteriori Probability

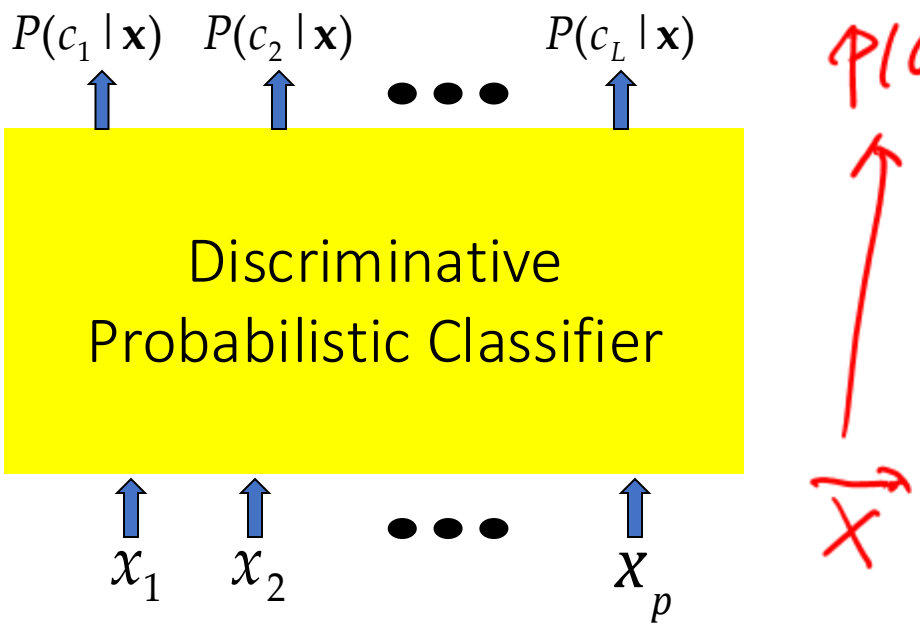
Please read extra slides for how to get MAP-rule from minimize EPE-0/1

Our Whole Section 2:

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

- Discriminative Classifiers

$$\arg\max_{c \in C} P(c / \mathbf{X}), \text{ where } C = \{c_1, \ldots, c_L\}$$



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

$\vec{x}$

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

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)

$$\log\left[\frac{P(\text{Head}|x)}{P(\text{Tail}|x)}\right]$$

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

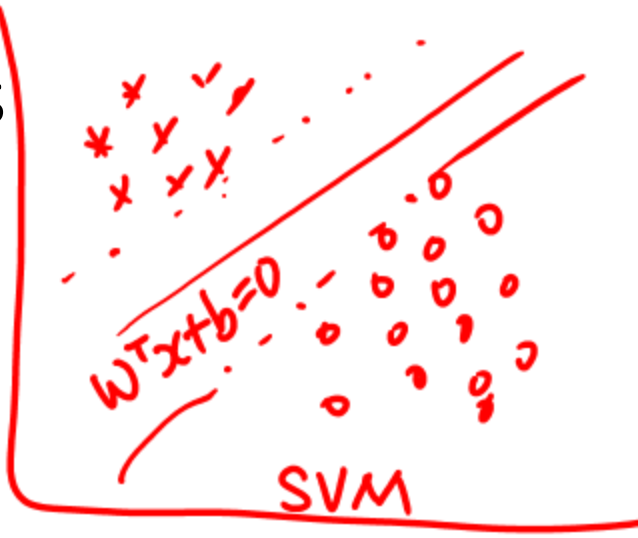
Logit

$p(c|x)$

$p(\text{Head})$

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

View IV: Logistic Regression models a linear classification boundary!



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

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

$$p(y=0|x) = p(y=1|x) \Rightarrow \text{Decision Boundary}$$

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

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

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

(conditional) Log-likelihood:

How?

For Bernoulli distribution

$$p(y \mid x)^y (1-p)^{1-y}$$

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

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

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

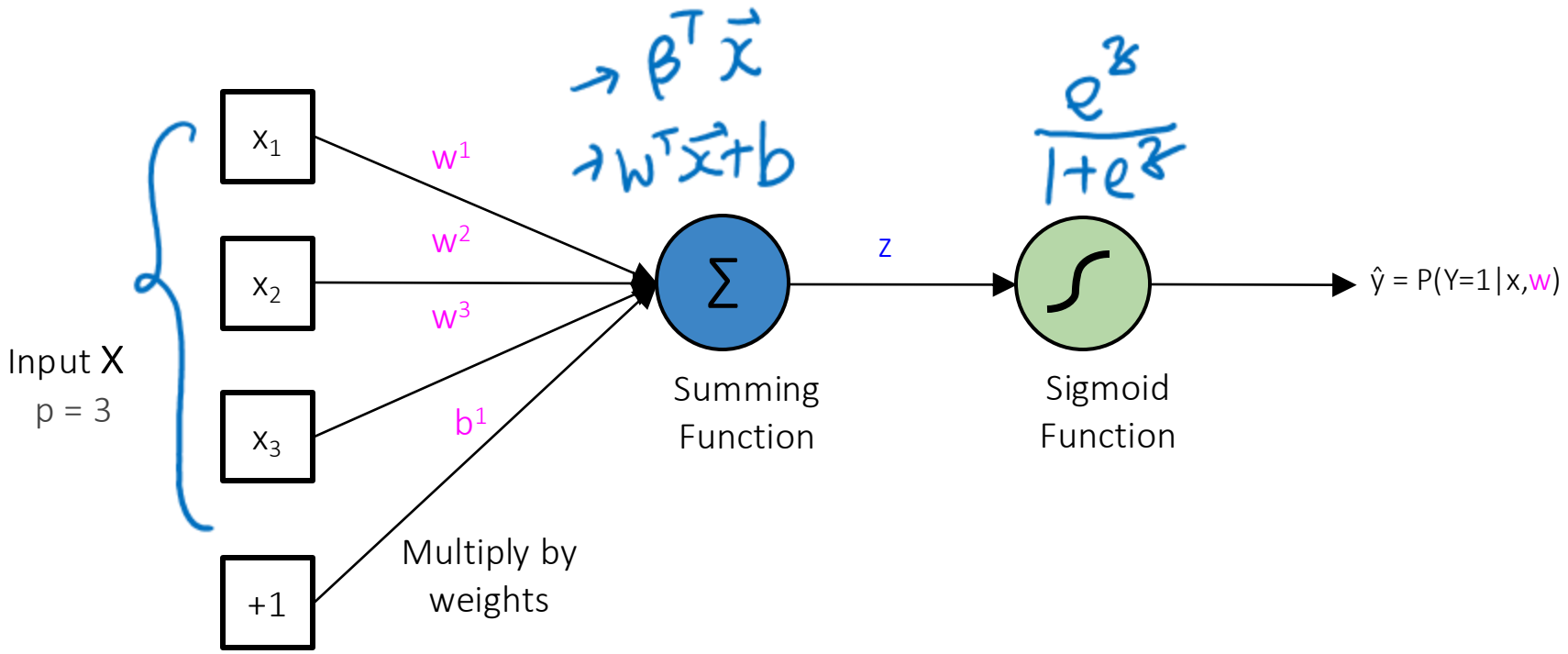
$$= \sum_{i=1}^{N} (y_i \beta^T x_i - \log(1+\exp(\beta^T x_i)))$$

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

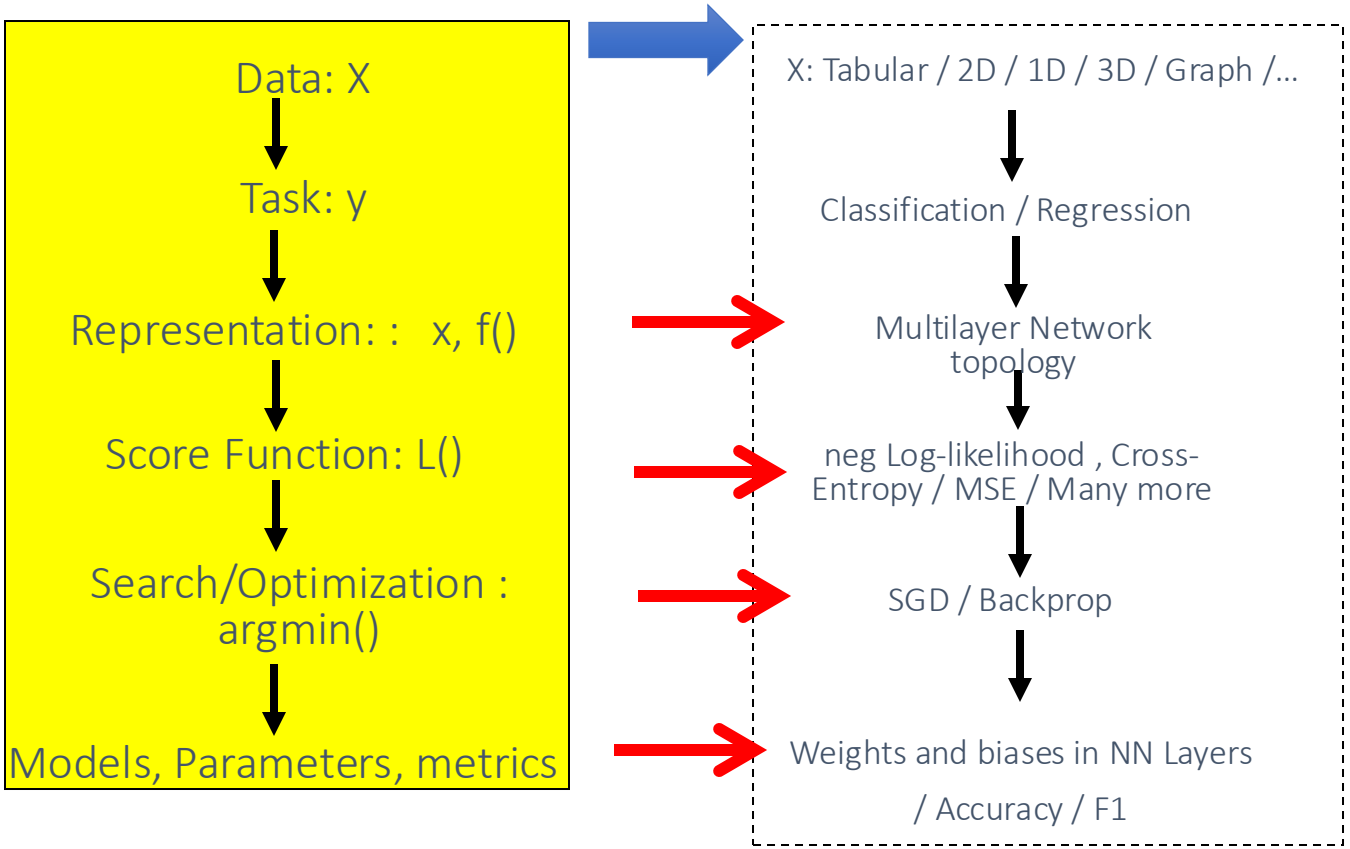
- HW3 is due
- Please select your Project's Shark Tank Sessions ASAP

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

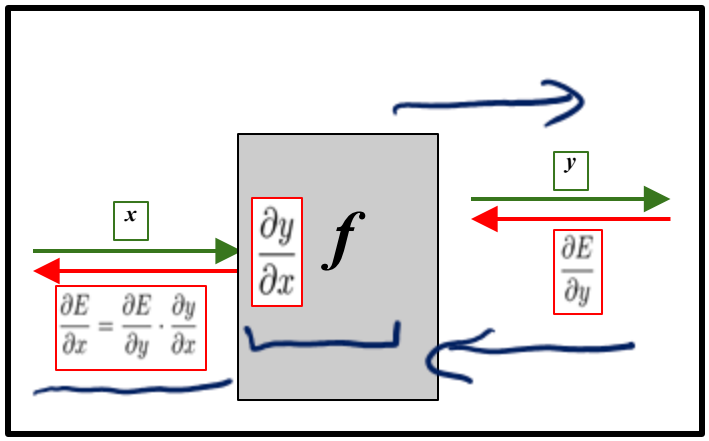
Takeaway: Logistic Regression **Classifier**

- 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

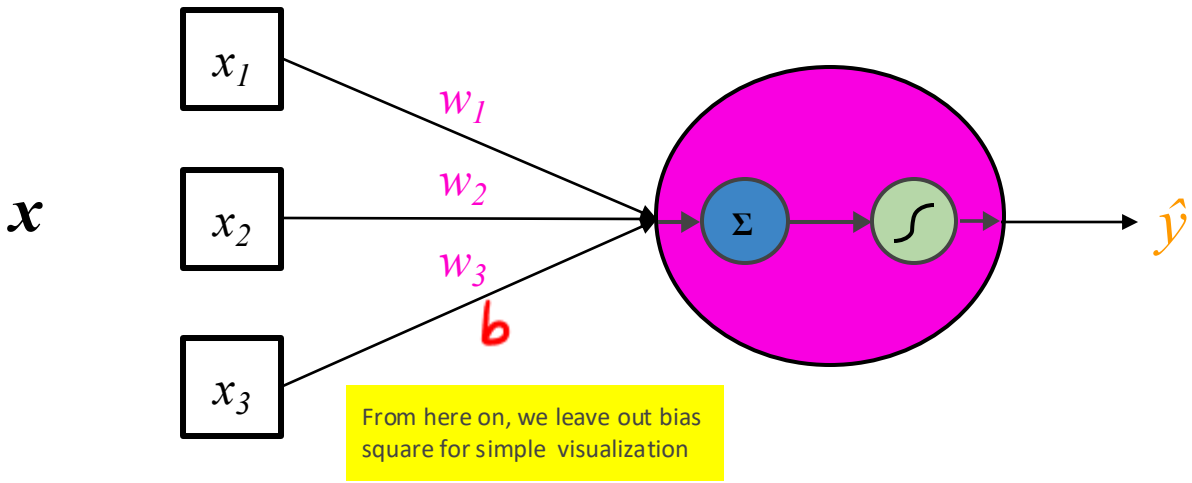


Building Deep Neural Nets



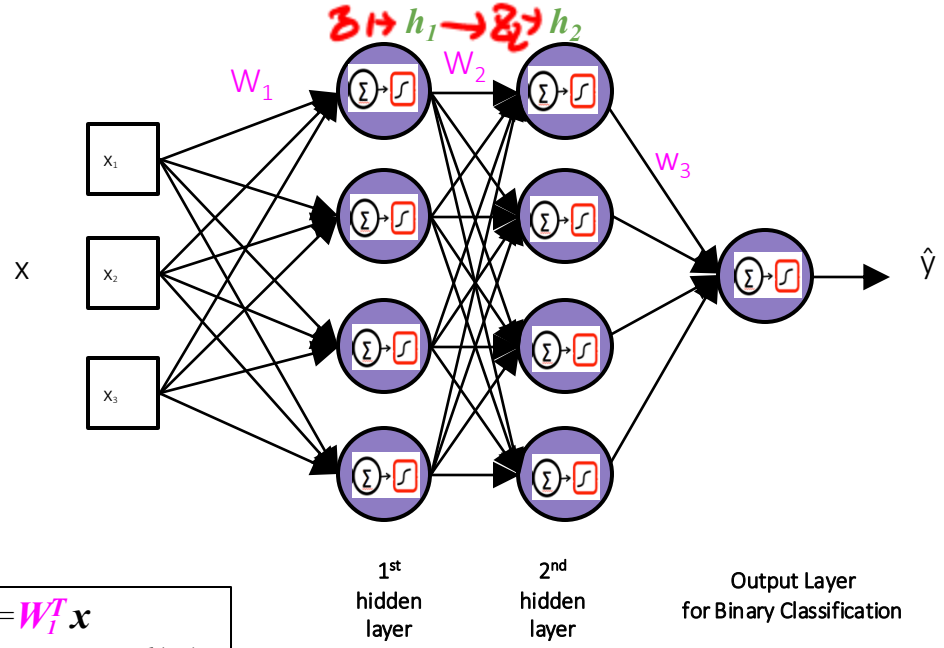
http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Neuron Representation



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

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

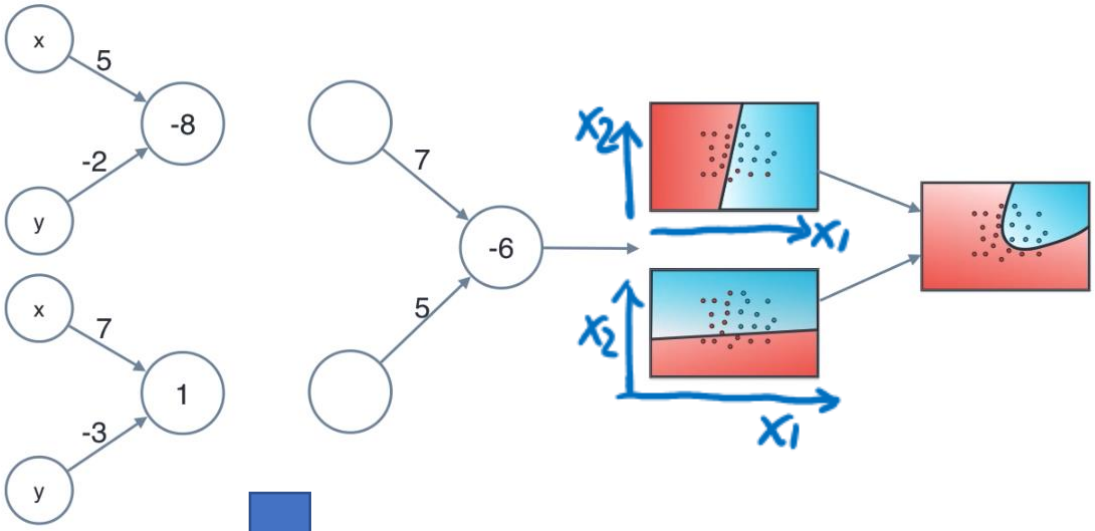


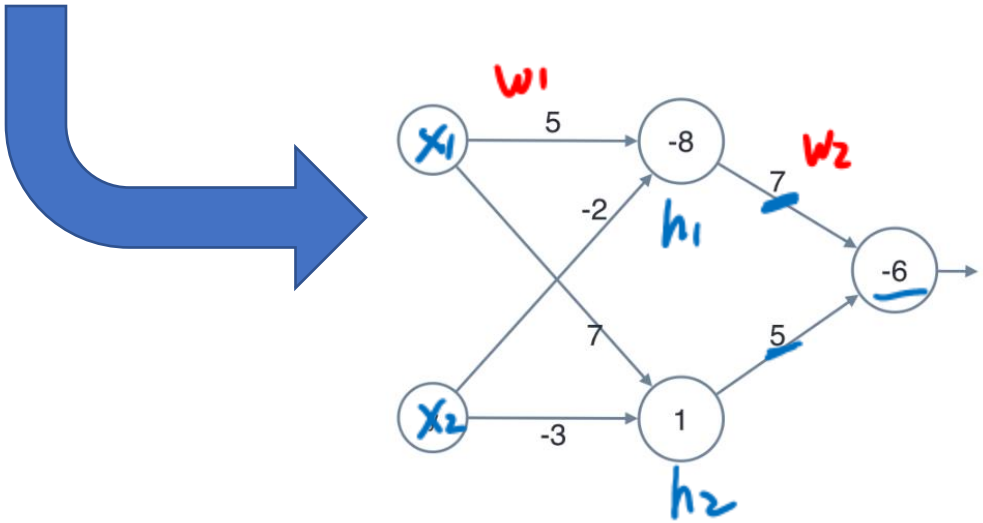
hidden layer 1 output

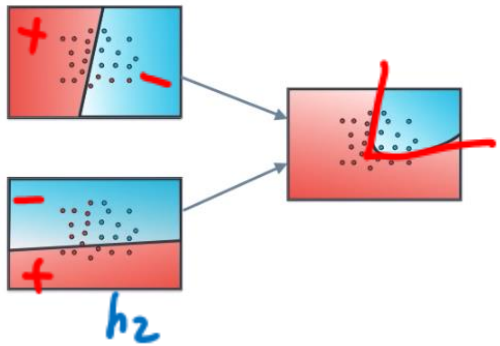
hidden layer 2 output

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

$X \xrightarrow{W_1} z_1 \to h_1 \xrightarrow{W_2} z_2 \to h_2 \xrightarrow{W_3} \hat{y}$

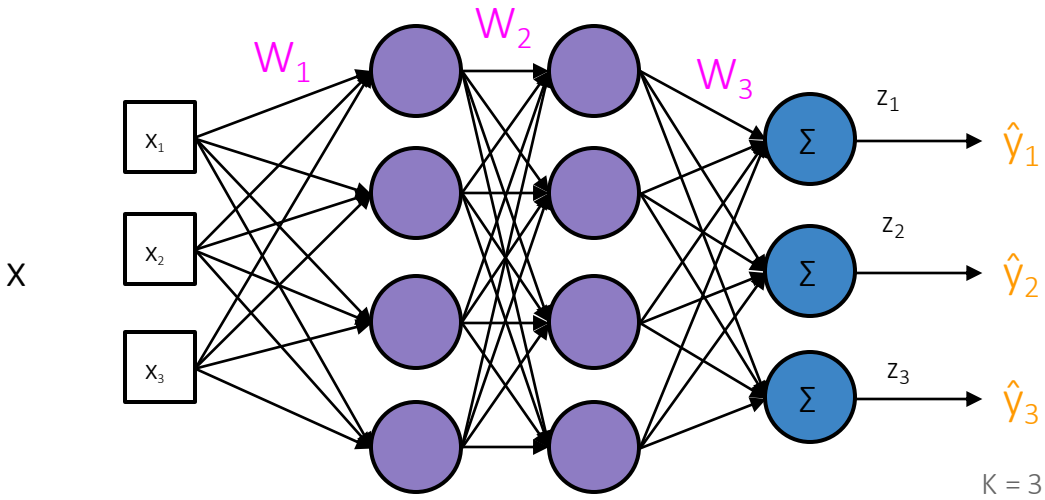

x
y
x
y
5
-2
7
-3
-8
1
7
5
-6
x_2
x_1
x_2
x_1


w_1
x_1
x_2
5
-2
7
-3
-8
1
h_1
h_2
w_2
7
5
-6


+
−
−
+
h_2

Recap: Multi-Class Classification Loss

Cross Entropy Loss



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

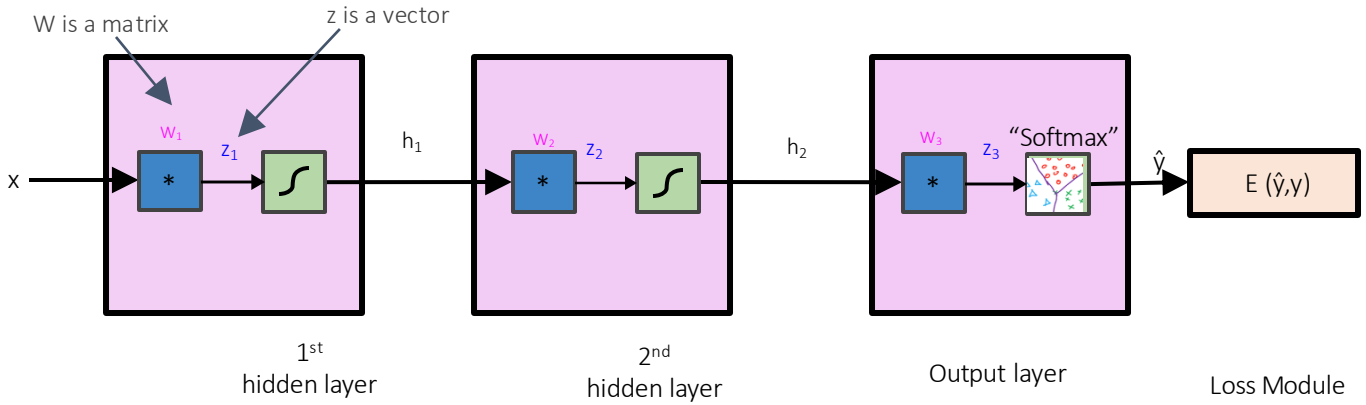
"Softmax" function.
Normalizing function which converts each class output to a probability.

$$y = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad \hat{y} = \begin{pmatrix} 0.1 \\ 0.7 \\ 0.2 \end{pmatrix}$$

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

"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

$$\textit{argmin\_w} \quad \{ f_4 (f_3 (f_2 (f_1 (\ )))) \}$$

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	$\frac{\partial z_1}{\partial x_1} = w_1$
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	$\partial E / \partial \hat{y} = -2(y - \hat{y})$

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

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

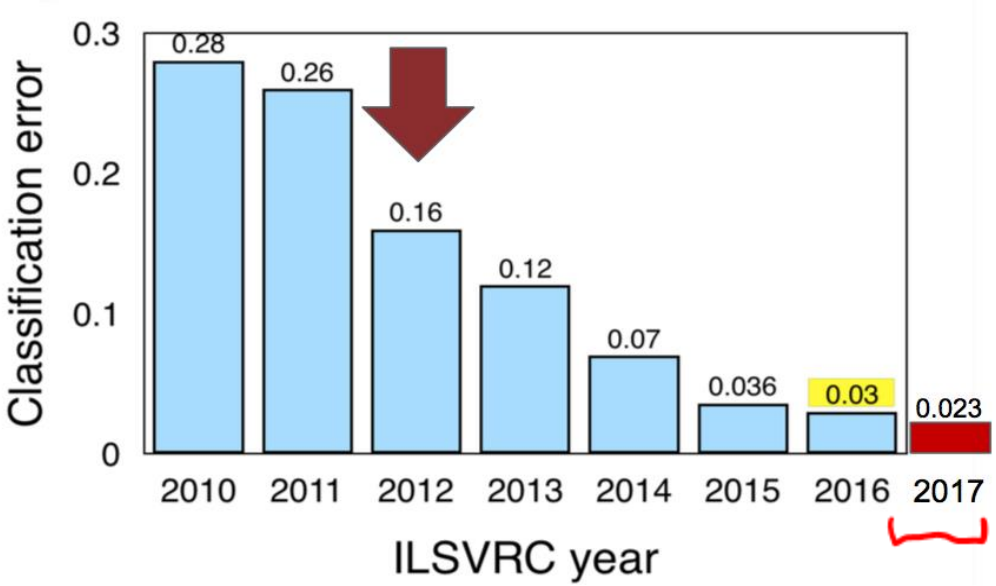
$$= -2(y - \hat{y}) \left(w_5 \frac{\partial h_1}{\partial w_1} + w_6 \frac{\partial h_2}{\partial w_1} \right)$$

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

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

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

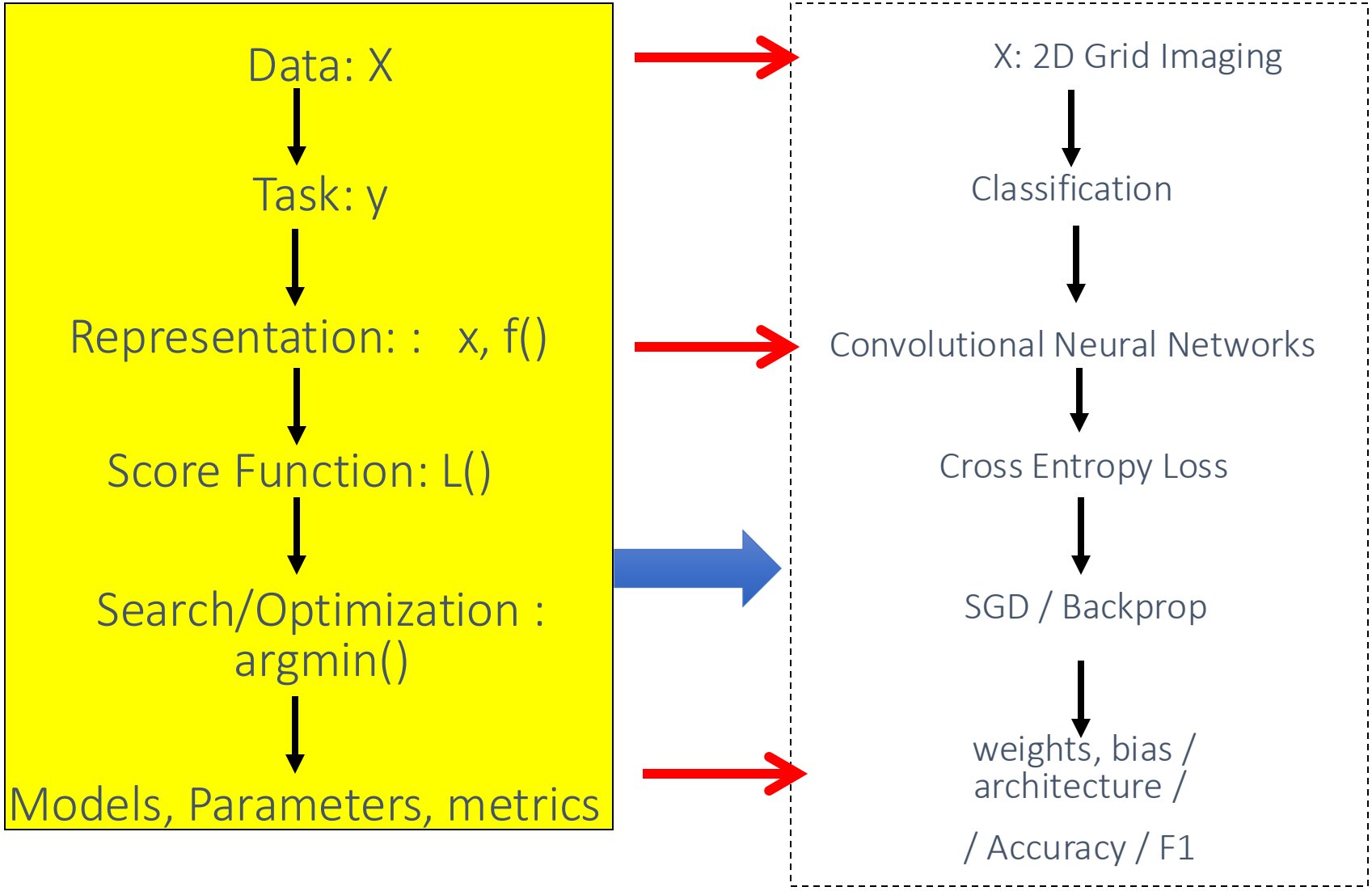


Adapt from From NIPS 2017 DL Trend Tutorial

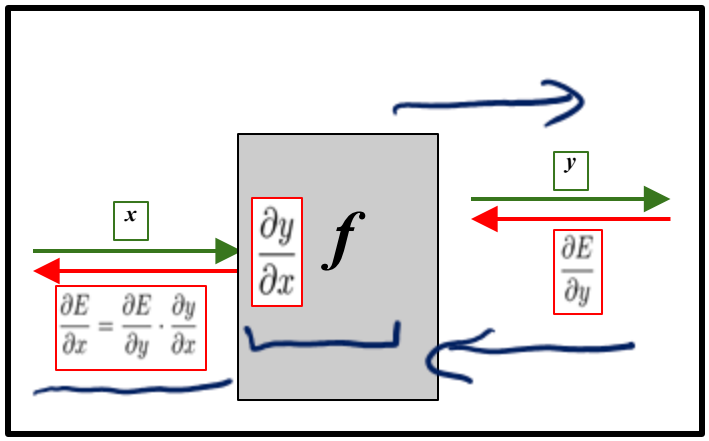
Metric	Formula	Interpretation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Overall performance of model
Precision	$\frac{TP}{TP + FP}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{TP}{TP + FN}$	Coverage of actual positive sample
Specificity	$\frac{TN}{TN + FP}$	Coverage of actual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

	actual	
	+	−
predicted+	*TP*	*FP*
predicted−	*FN*	*TN*

Today: Convolutional Network Models on 2D Grid / Image



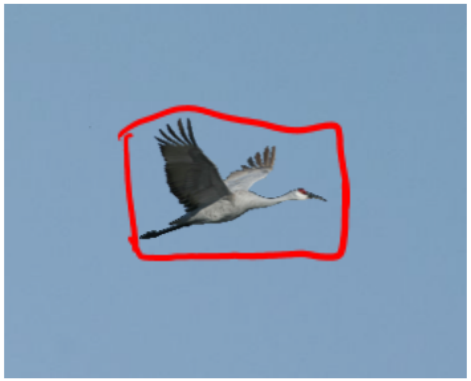
Building Deep Neural Nets



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

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!

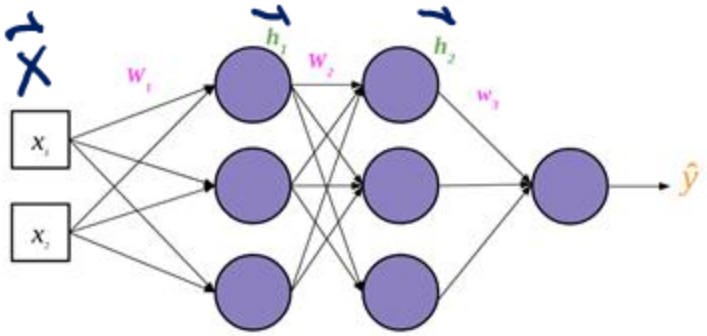
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

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




```python
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

Convolution

Max Pooling

A new image

Convolution

Max Pooling

A new image

Can repeat many times

Flatten

Fully Connected Feedforward network

softmax

cat dog ……

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

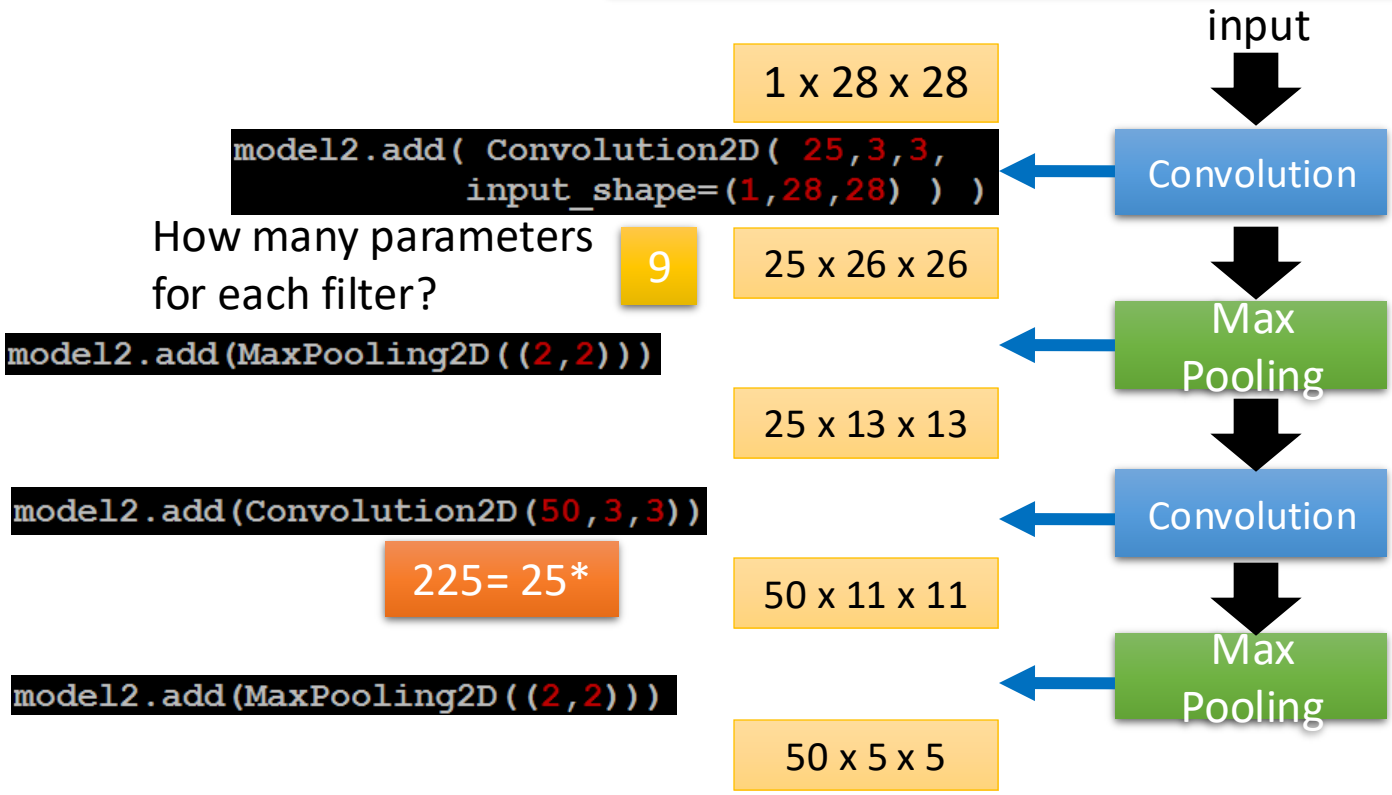
Property 3

- Subsampling the pixels will not change the object

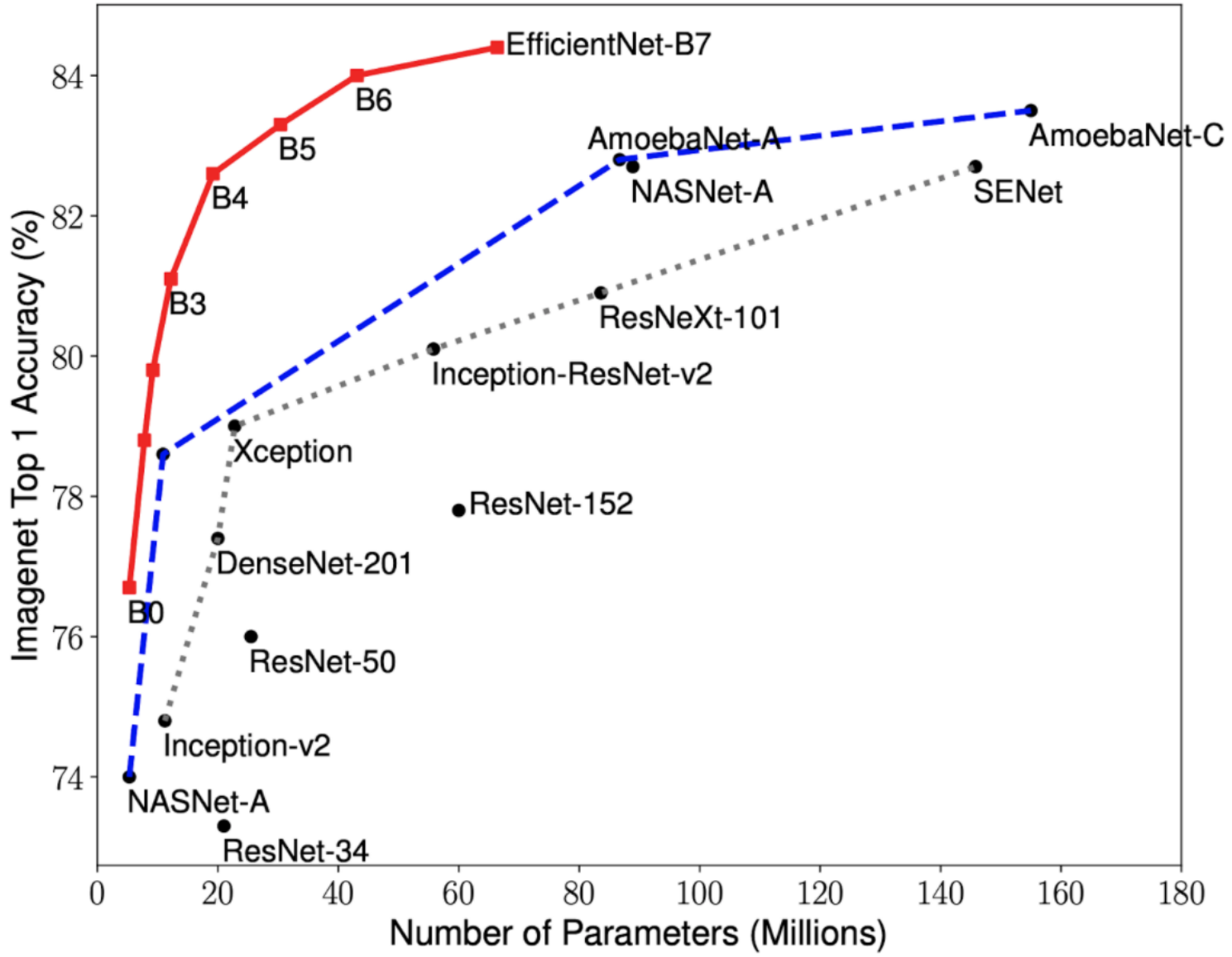
Adapted Dr. Hung-yi Lee's CNN slides

CNN in Keras

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



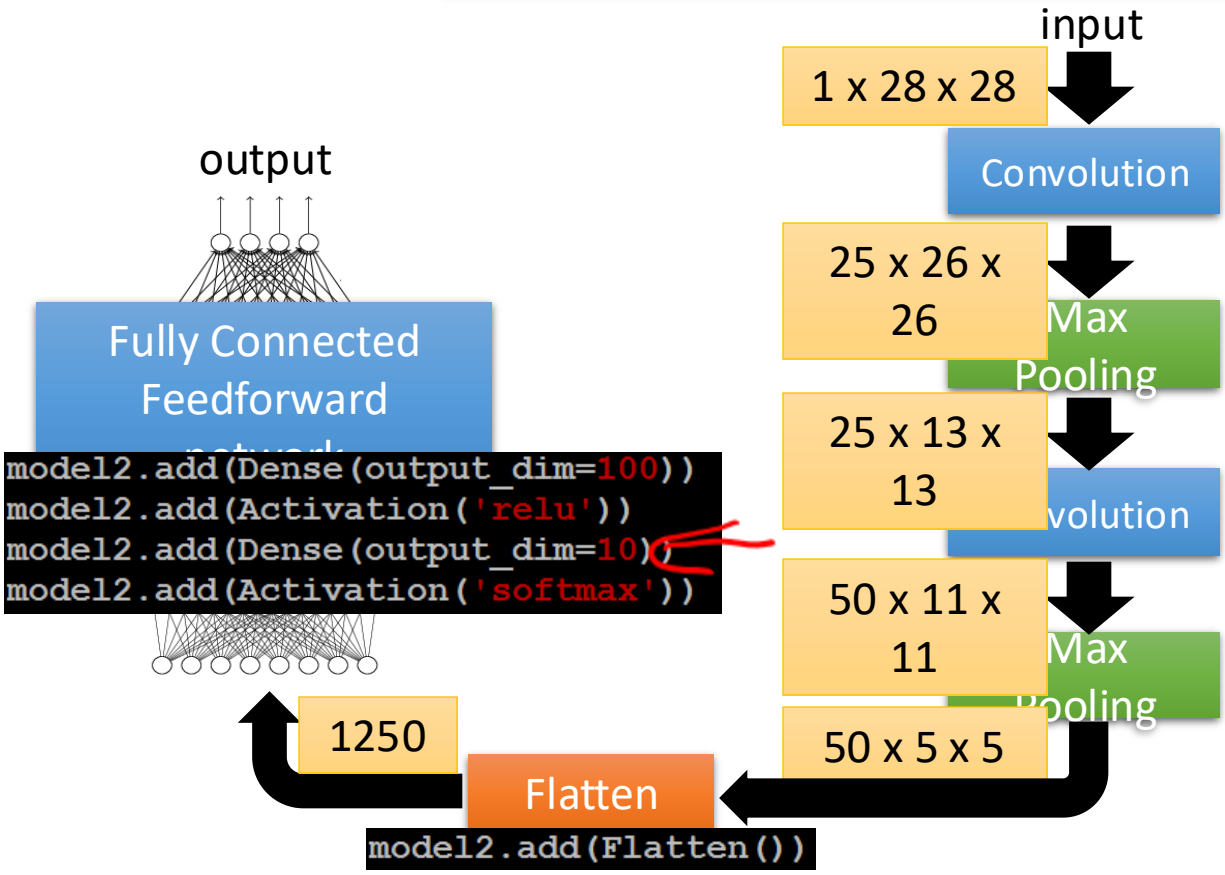
Dr. Hung-yi Lee's CNN slides



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)

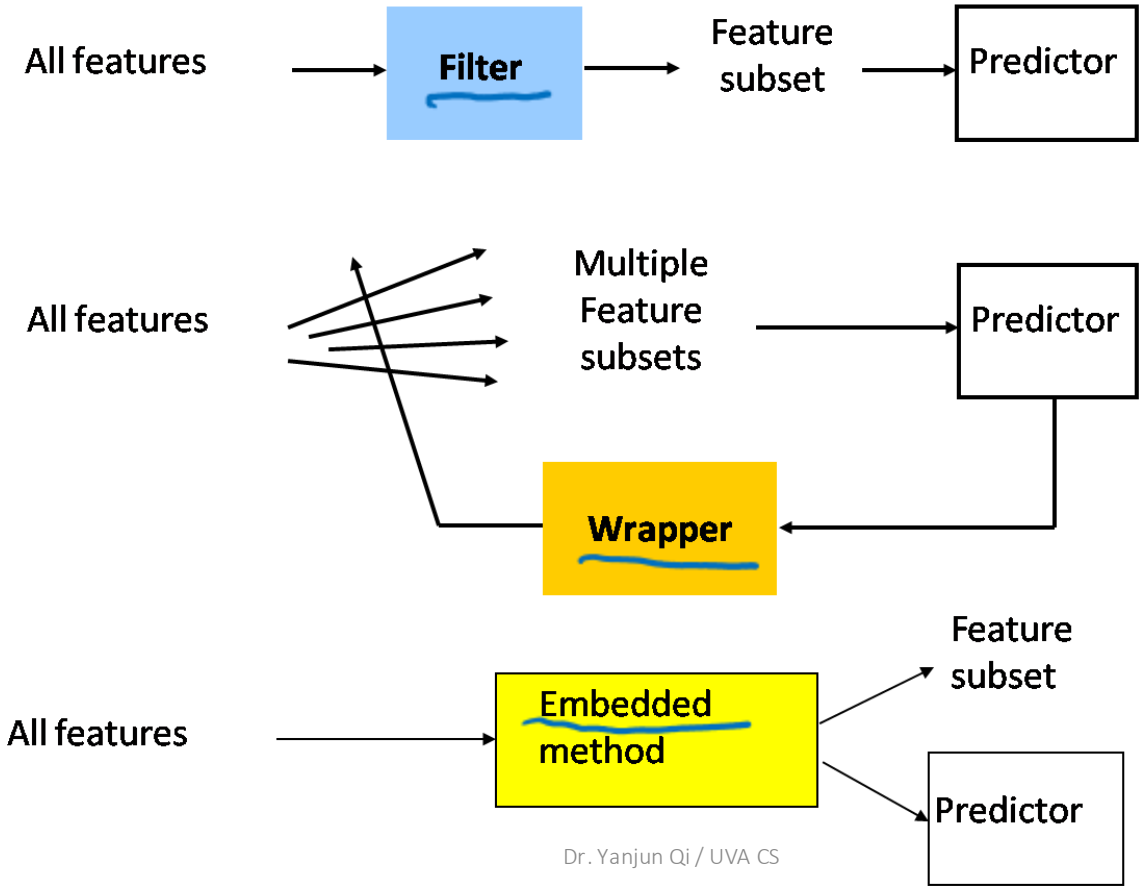
Feature selection: chooses a subset of the original features.

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

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.

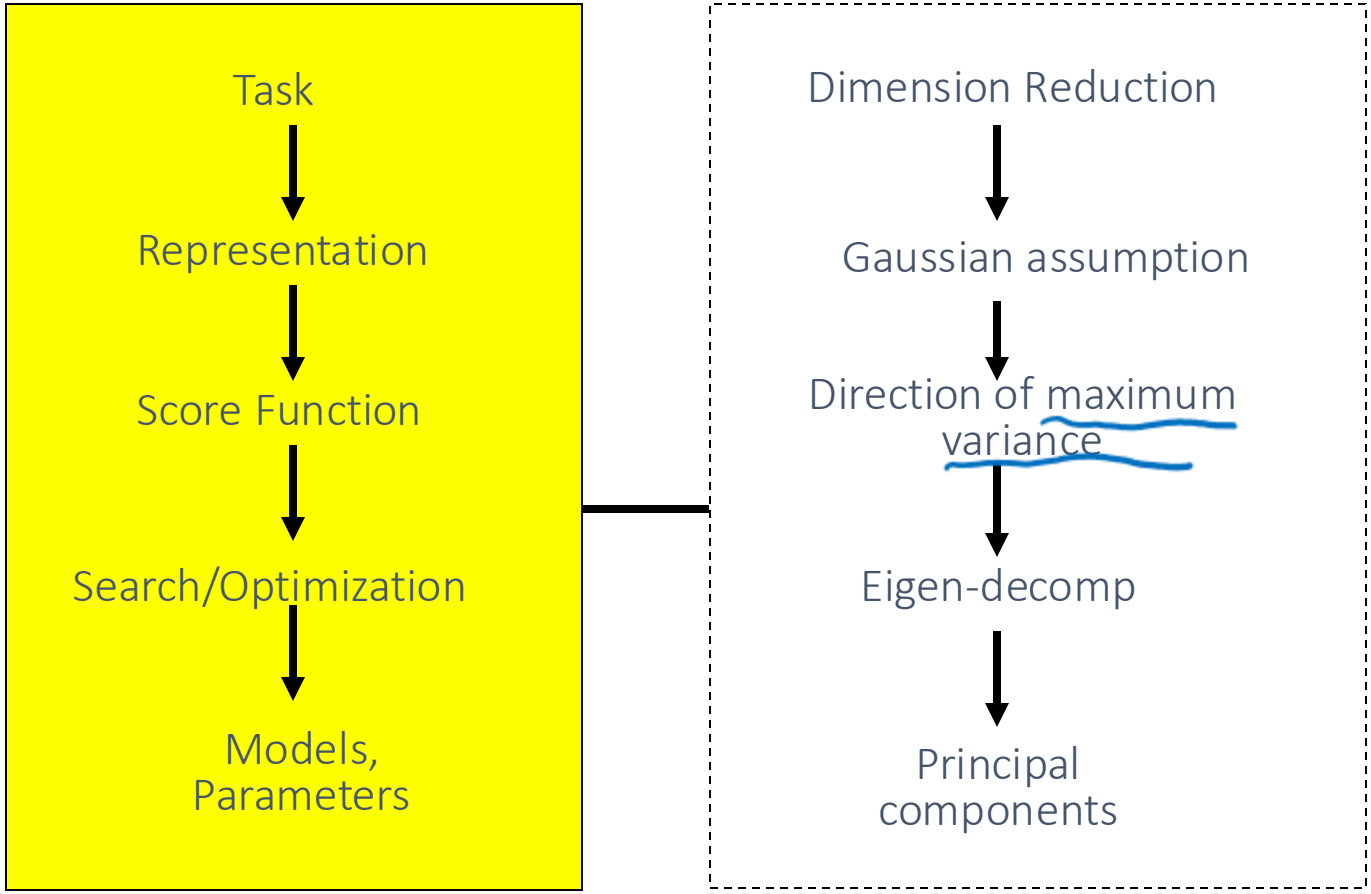
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \xrightarrow{U^T} \mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_K \end{bmatrix}$$

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.

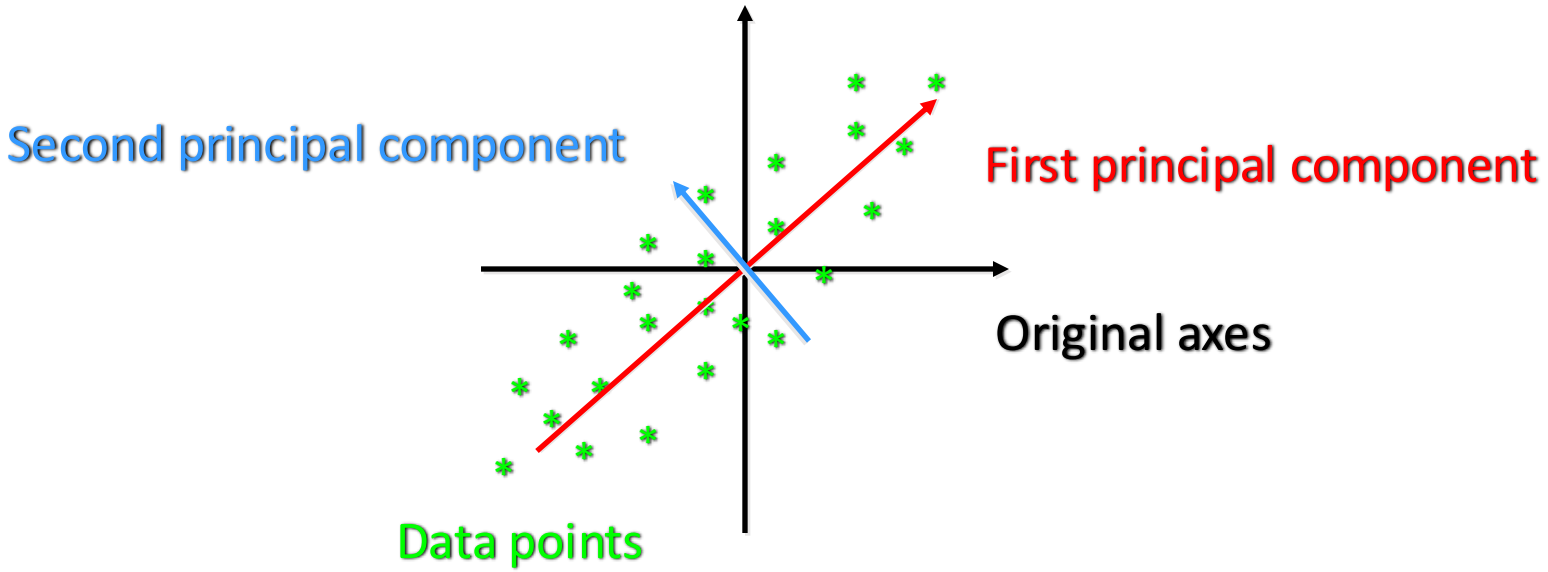
- Recent nonlinear feature extraction methods:
  - Like Word Embedding / Autoencoder / …

Principal Component Analysis

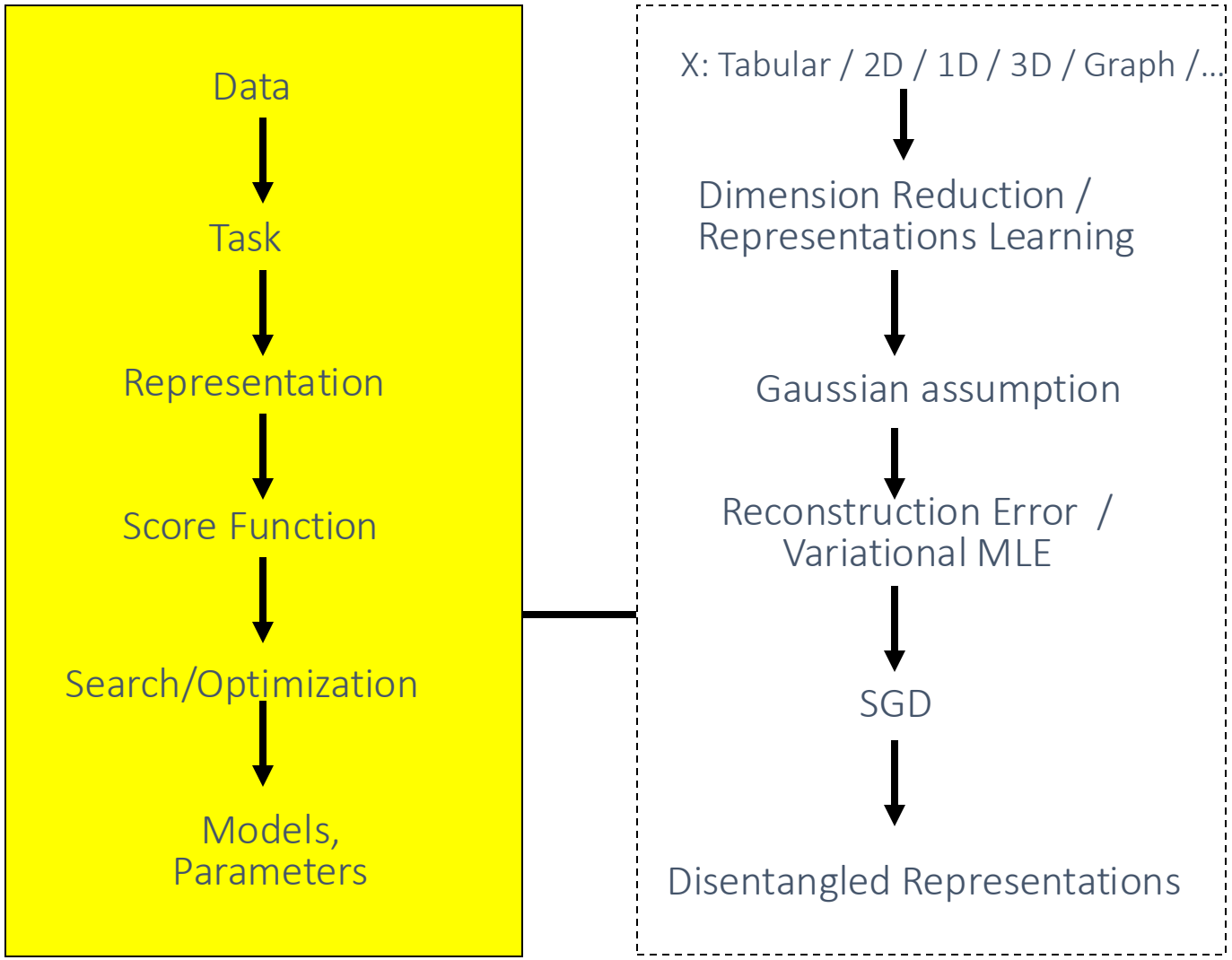


How does PCA work? Explaining Variance

- 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.

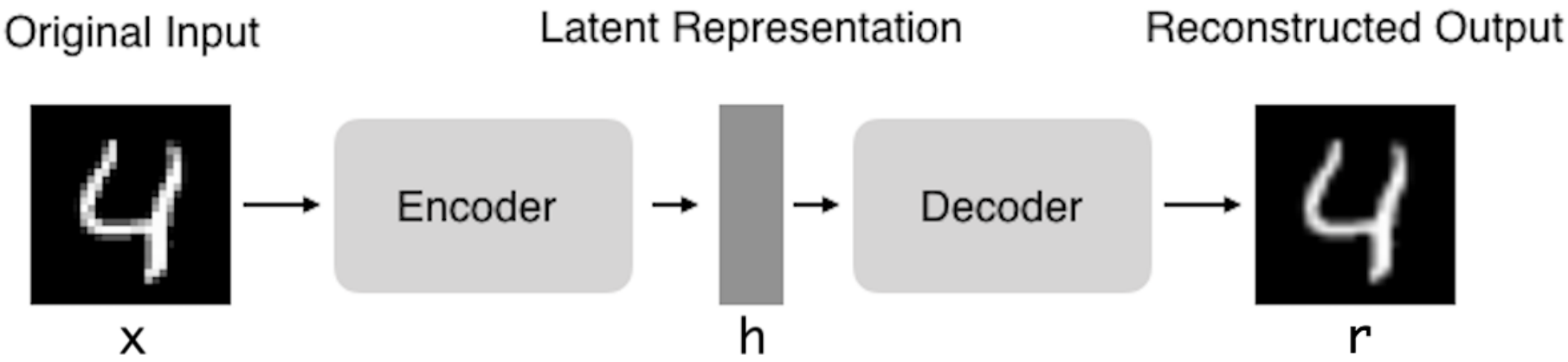


Auto Encoder



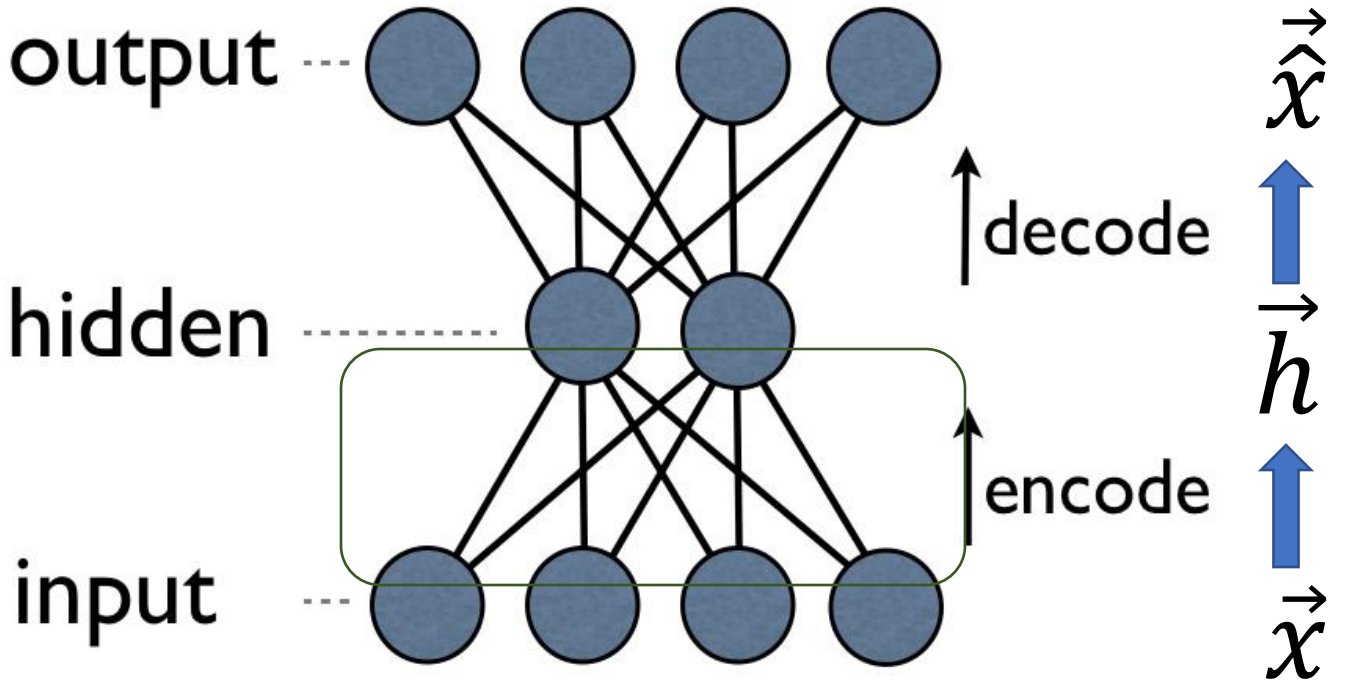
Autoencoders: structure

- 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

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

- Basic approach – "one hot vector"
  - Binary vector
  - Length = | vocab |
  - 1 in the position of the word id, the rest are 0
  - However, does not represent word meaning
  - Extremely high dimensional (there are over 200K words in the English language)
  - Extremely sparse

- Solution:
  Distributional Word Embedding Vectors

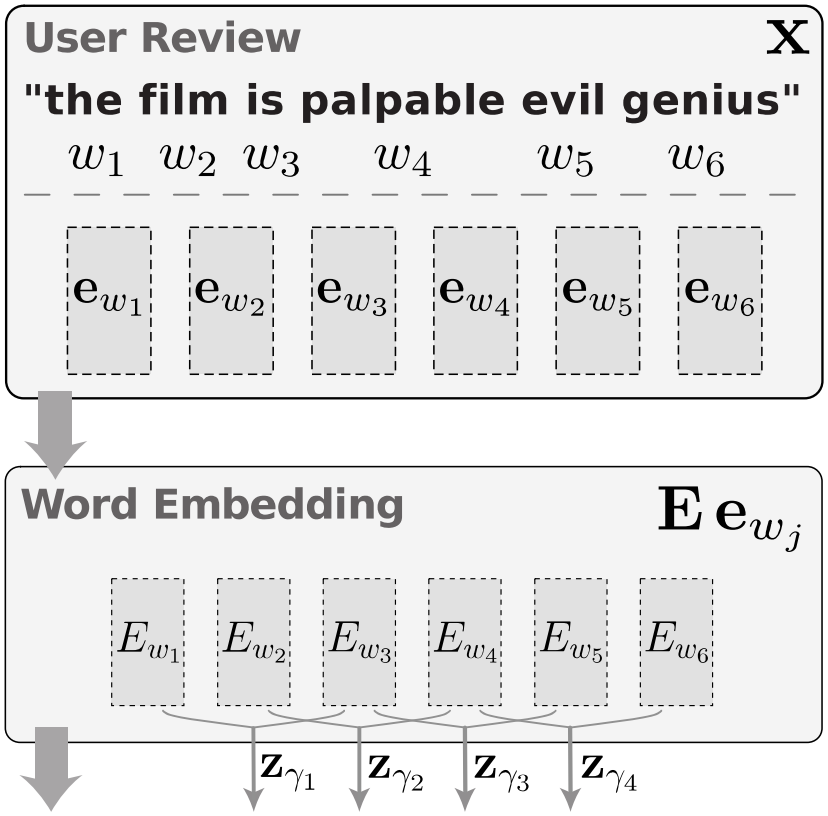
Popular word embeddings

- GloVe (Global Vectors)
  - Pennington et al., 2014
- 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



S3: Lecture 18:

Deep Neural Networks for Natural Language Processing