UVA CS 6316 – Fall 2015 Graduate: Machine Learning

Lecture 22: Review

Dr. Yanjun Qi

University of Virginia

Department of
Computer Science

Dr. Yanjun Qi / UVA CS 6316 / f15

Announcements: Rough Plan

- HW5
  - Out on Nov. 18th
  - Due on Dec. 7th
- Project Presentation
  - Due on Dec. 2nd midnight
  - Presentations @ Dec 3rd and Dec 4th
- Project Final Report
  - Due on Dec. 11th midnight

Announcements: Exam

- Open Note / Open Lectures
- No laptop / No Cell phone / No internet access / No electronic devices

- Covering contents till today
  - Practice with sample questions in HW4
  - HW4 due on Nov. 20th
  - Please review course slides carefully

Today

- ❑ History of AI & Machine Learning
- ❑ Review of ML methods covered so far

What are the goals of AI research?

Artifacts that THINK like HUMANS

Artifacts that THINK RATIONALLY

Artifacts that ACT like HUMANS

Artifacts that ACT RATIONALLY

From: M.A. Papalaskar

Dr. Yanjun Qi / UVA CS 6316 / f15

A Bit of History

1940s

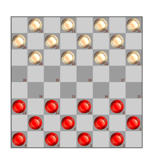
Advances in mathematical logic, information theory, concept of neural computation

- 1943: McCulloch & Pitts Neuron
- 1948: Shannon: Information Theory
- 1949: Hebbian Learning
  - cells that fire together, wire together

1950s

Early computers. Dartmouth conference coins the phrase “artificial intelligence” and Lisp is proposed as the AI programming language

- 1950: Turing Test
- 1956: Dartmouth Conference
- 1958: Friedberg: Learn Assembly Code
- 1959: Samuel: Learning Checkers



From: M.A. Papalaskar

1960s

A.I. funding increased (mainly military). Famous quote: "Within a generation ... the problem of creating 'artificial intelligence' will substantially be solved."

Early symbolic reasoning approaches.

- Logic Theorist, GPS, Perceptrons

- 1969: Minsky & Papert "Perceptrons"

1970s

A.I. "winter" – Funding dries up as people realize this is a hard problem!

Limited computing power and dead-end frameworks lead to failures.

- eg: Machine Translation Failure

From: M.A. Papalaskar

1980s

Rule based "expert systems" used in medical / legal professions.

Bio-inspired algorithms (Neural networks, Genetic Algorithms).

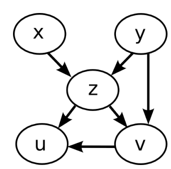
Again: A.I. promises the world – lots of commercial investment

Expert Systems (Mycin, Dendral, EMYCIN

Knowledge Representation and reasoning:

Frames, Eurisko, Cyc, NMR, fuzzy logic

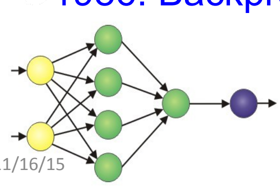
Speech Recognition (HEARSAY, HARPY, HWIM)



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

Machine Learning:

- 1982: Hopfield Nets, Decision Trees, GA & GP.
- 1986: Backpropagation, Explanation-Based Learning



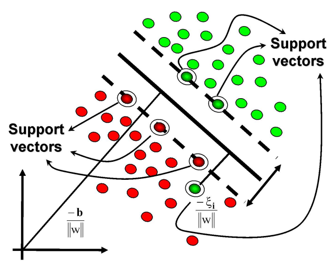
From: M.A. Papalaskar

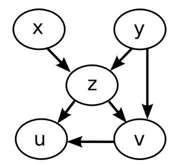
1990s

Some concrete successes begin to emerge. AI diverges into separate fields: Computer Vision, Automated Reasoning, Planning systems, Natural Language processing, **Machine Learning**…

…Machine Learning begins to overlap with statistics / probability theory.

- 1992: Koza & Genetic Programming
- 1995: Vapnik: Support Vector Machines





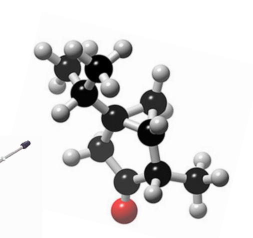
$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

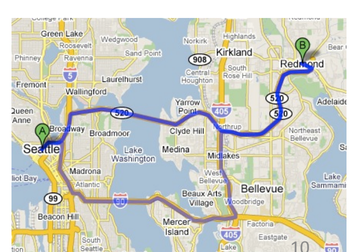
From: M.A. Papalaskar

2000s

First commercial-strength applications: Google, Amazon, computer games, route-finding, credit card fraud detection, spam filters, etc…

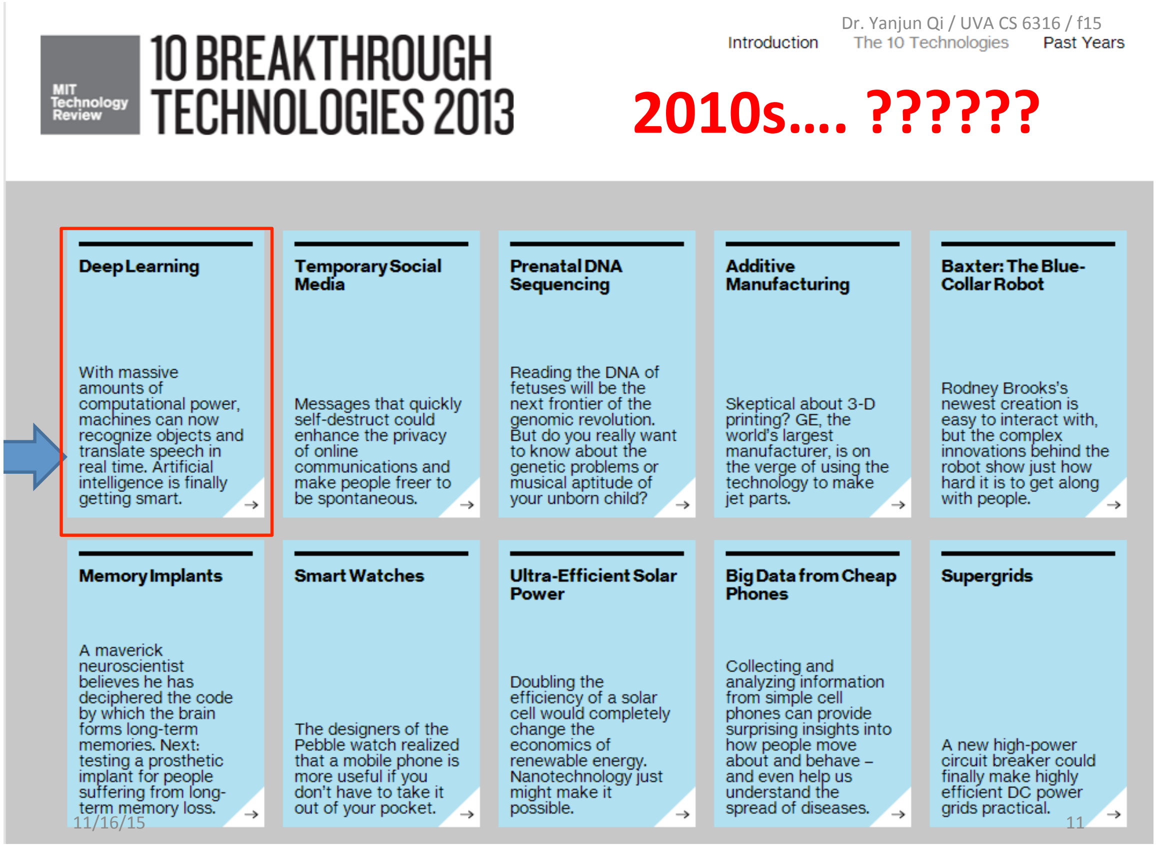
Tools adopted as standard by other fields e.g. biology





From: M.A. Papalaskar

2010s.... ??????



How can we build more intelligent computer / machine ?

- Able to
  - **perceive the world**
  - **understand the world**
- This needs
  - Basic speech capabilities
  - Basic vision capabilities
  - Language understanding
  - User behavior / emotion understanding
  - Able to think ??

Plenty of Data

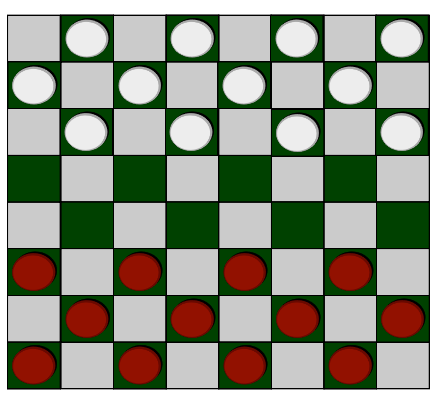
- Text: trillions of words of English + other languages
- Visual: billions of images and videos
- Audio: thousands of hours of speech per day
- User activity: queries, user page clicks, map requests, etc,
- Knowledge graph: billions of labeled relational triplets
- ....

Data-driven machine learning methods have made machines / computers much more intelligent

Dr. Jeff Dean's talk

Samuel's definition of ML (1959)

- Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.



Tom Mitchell (1998): Well-posed Learning Problem

*A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**.*

Defining the Learning Task

Improve on task, T, with respect to performance metric, P, based on experience, E.

T: Playing checkers
P: Percentage of games won against an arbitrary opponent
E: Playing practice games against itself

T: Recognizing hand-written words
P: Percentage of words correctly classified
E: Database of human-labeled images of handwritten words

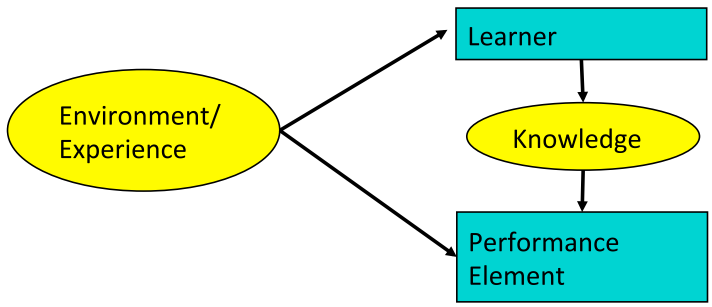
T: Driving on four-lane highways using vision sensors
P: Average distance traveled before a human-judged error
E: A sequence of images and steering commands recorded while observing a human driver.

T: Determine which students like oranges or apples
P: Percentage of students' preferences guessed correctly
E: Student attribute data

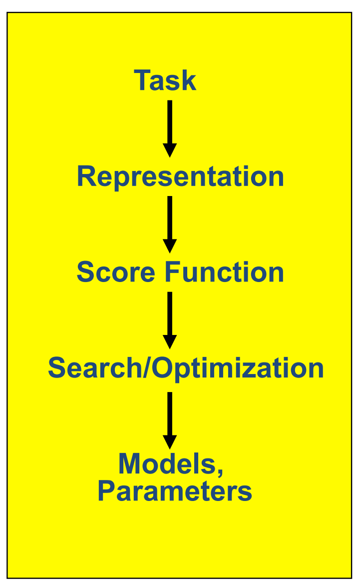
From: M.A. Papalaskar

Designing a Learning System

- Choose the ***training experience***
- Choose exactly what is to be learned, i.e. the ***target function***.
- Choose a ***learning algorithm*** to infer the target function from the experience.
- A learning algorithm will also determine a ***performance measure***



Machine Learning in a Nutshell



ML grew out of work in AI

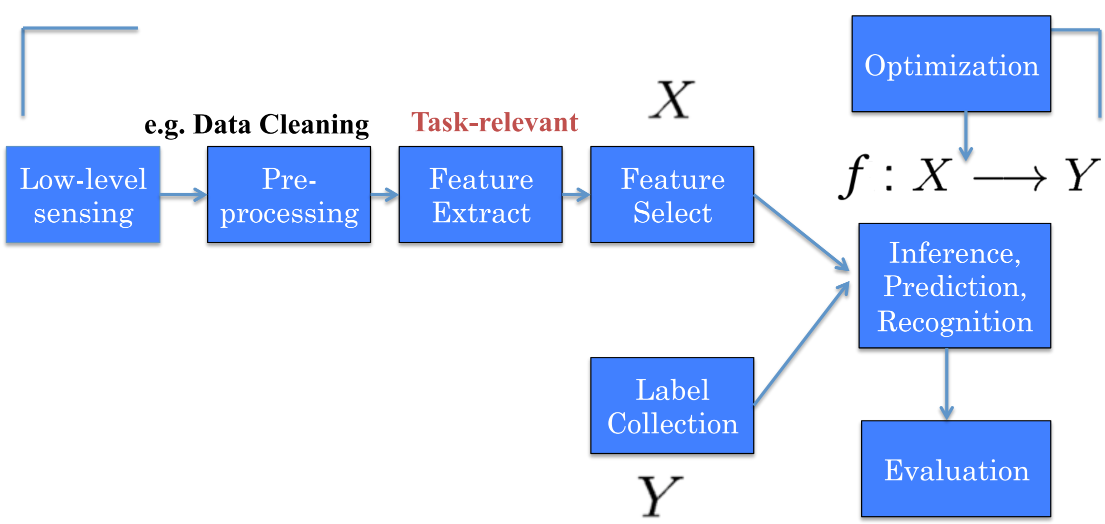
Optimize a performance criterion using example data or past experience,

Aiming to generalize to unseen data

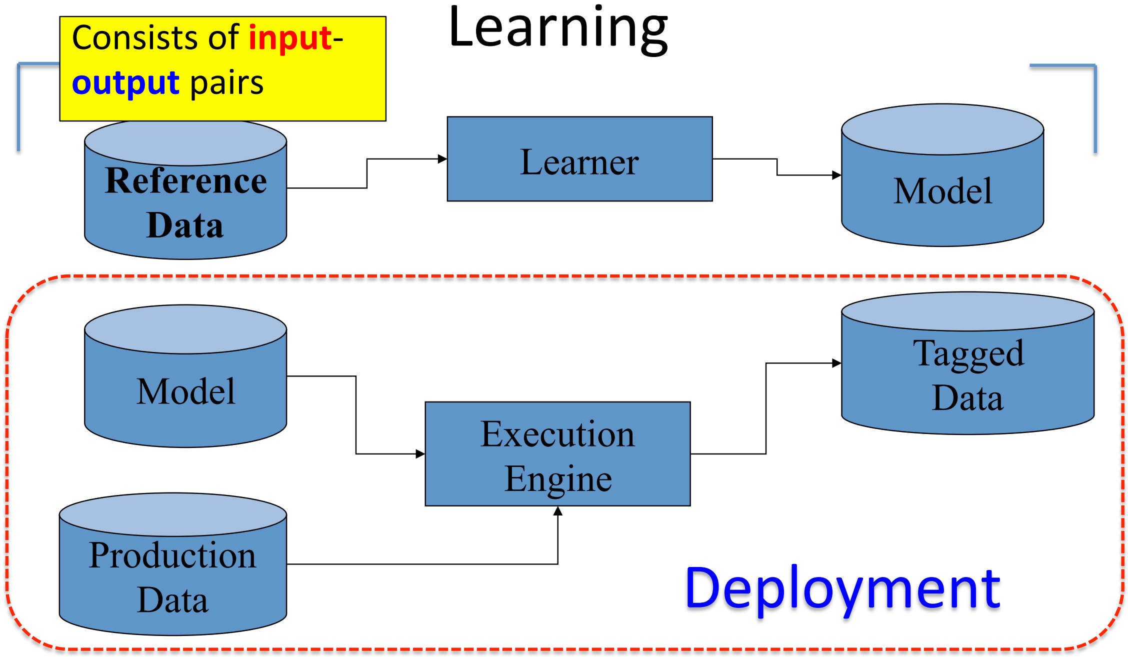
What we have covered for each

Task	
Representation	
Score Function	
Search/ Optimization	
Models, Parameters	

A Typical Machine Learning Pipeline



An Operational Model of Machine Learning



Today

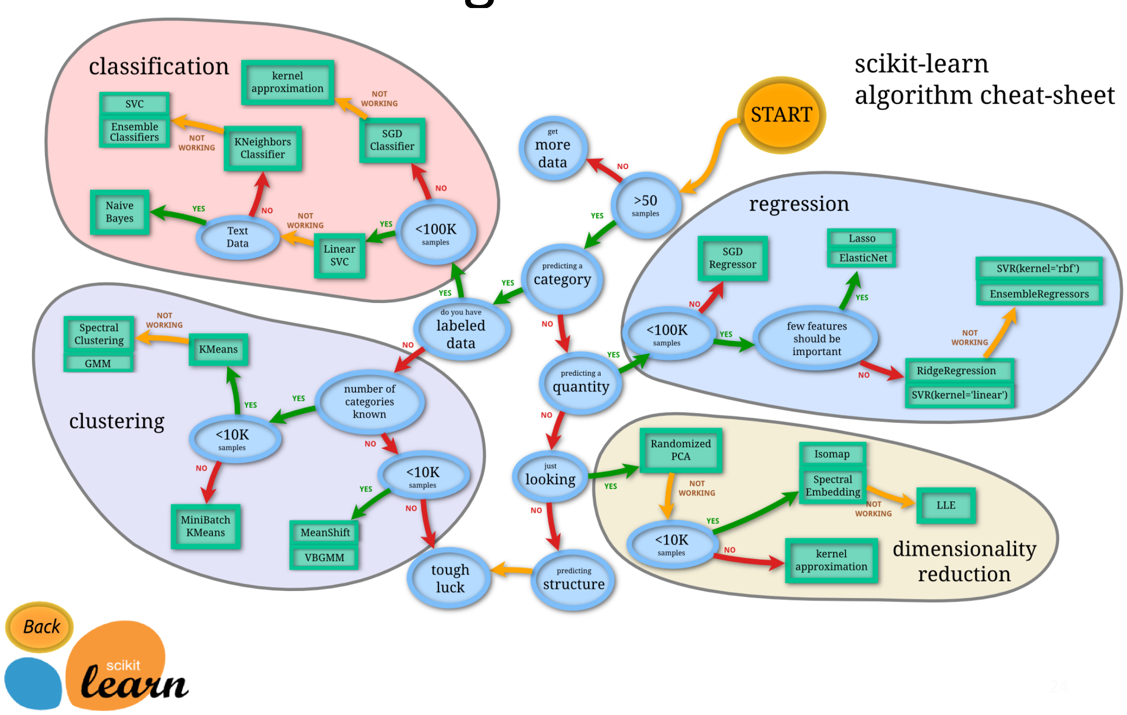
- ❑ History of Machine Learning & AI
- ❑ Review of ML methods covered so far

Where are we ? ➔ major sections of this course

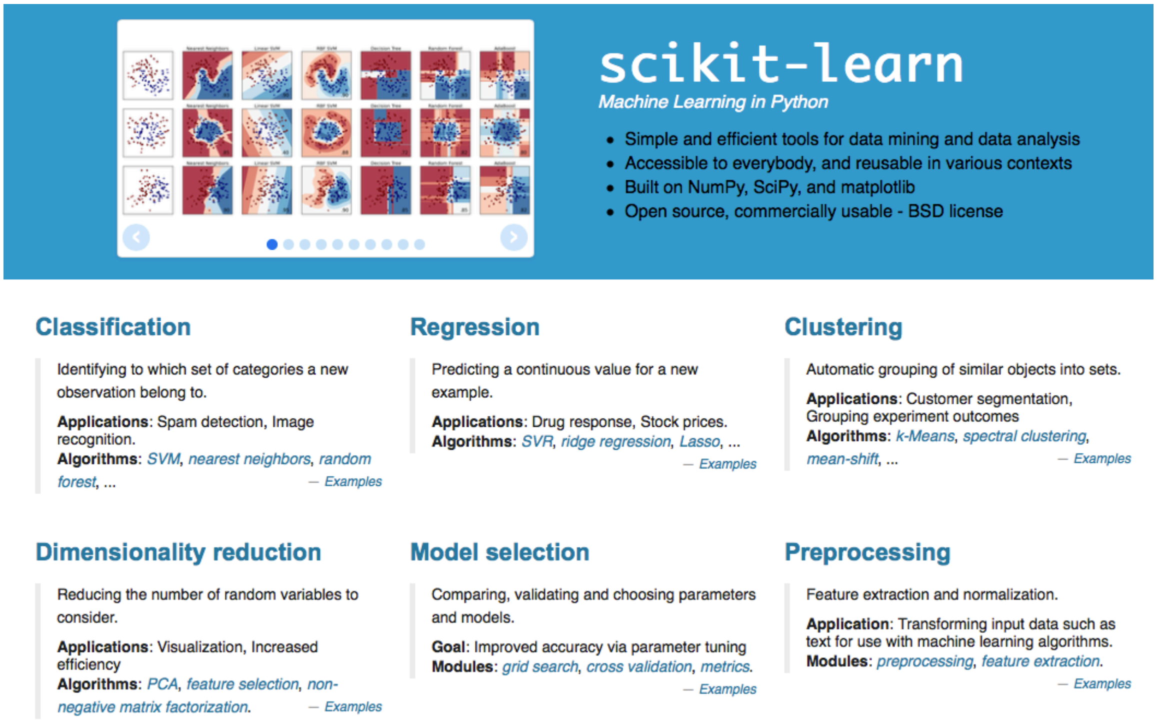
- ❑ Regression (supervised)
- ❑ Classification (supervised)
- ❑ Unsupervised models
- ❑ Learning theory
- ❑ Graphical models

http://scikit-learn.org/stable/tutorial/machine_learning_map/

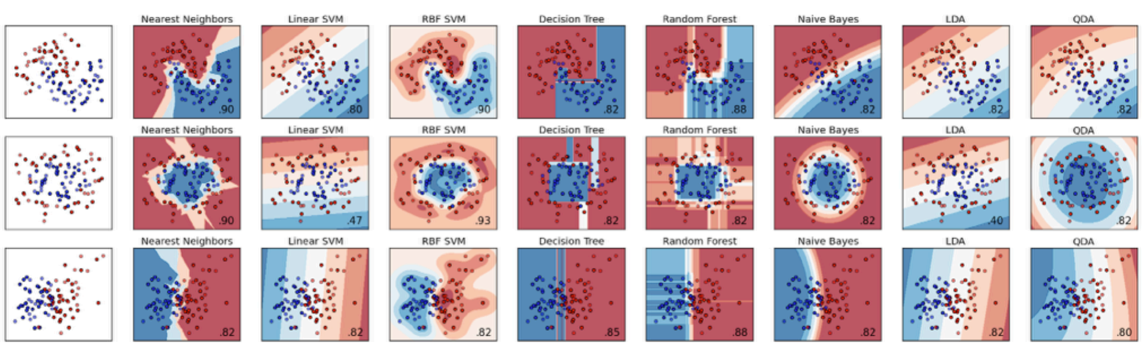
Scikit-learn algorithm cheat-sheet



http://scikit-learn.org/stable/



Dr. Yanjun Qi / UVA CS 6316 / f15



- ✓ different assumptions on data
- ✓ different scalability profiles at training time
- ✓ different latencies at prediction time
- ✓ different model sizes (embedability in mobile devices)

Olivier Grisel's talk

What we have covered (I)

- Supervised Regression models
  - Linear regression (LR)
  - LR with non-linear basis functions
  - Locally weighted LR
  - LR with Regularizations

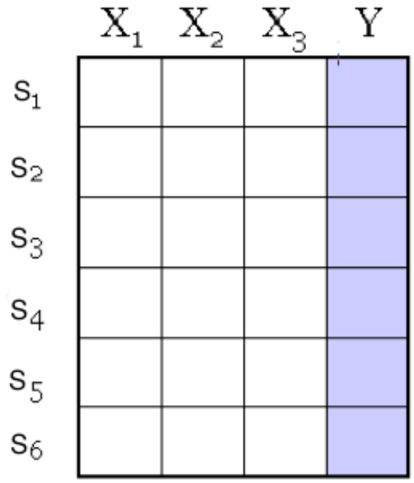
e.g. SUPERVISED LEARNING

$$f : X \longrightarrow Y$$

- Find function to map input space X to output space Y

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

KEY



A Dataset

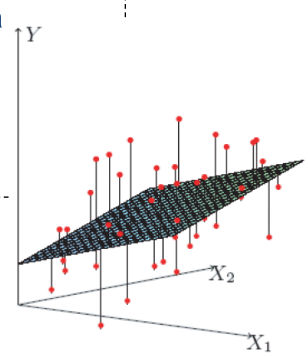
$$f : \boxed{X} \longrightarrow \boxed{Y}$$

Output Y as continuous values

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/ predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

(1) Multivariate Linear Regression

Task	Regression
↓	↓
Representation	Y = Weighted linear sum of X's
↓	↓
Score Function	Least-squares
↓	↓
Search/Optimization	Linear algebra
↓	↓
Models, Parameters	Regression coefficients



$$\hat{y} = f(x) = \theta_0 + \theta_1 x^1 + \theta_2 x^2 \Rightarrow \theta$$

Method I: normal equations

- Write the cost function in matrix form:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$$
$$= \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y})$$
$$= \frac{1}{2}\left(\theta^T X^T X\theta - \theta^T X^T\vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}\right)$$

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n^T & -- \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

To minimize $J(\theta)$, take derivative and set to zero:

$$\Rightarrow \boxed{X^T X\theta = X^T\vec{y}}$$

The normal equations

$$\Downarrow$$

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

Method II: LR with batch Steepest descent / Gradient descent

$$\theta_t = \theta_{t-1} - \alpha\nabla J(\theta_{t-1})$$

For the t-th epoch

$$\nabla_\theta J = \left[\frac{\partial}{\partial\theta_1}J, \ldots, \frac{\partial}{\partial\theta_k}J\right]^T = -\sum_{i=1}^{n}(y_i - \vec{\mathbf{x}}_i^T\theta)\mathbf{x}_i$$

$$\theta_j^{t+1} = \theta_j^{t} + \alpha\sum_{i=1}^{n}(y_i - \vec{\mathbf{x}}_i^T\theta^t)x_i^j$$

– This is as a **batch** gradient descent algorithm

Method III: LR with Stochastic GD ➔

- From the batch steepest descent rule:

$$\theta_j^{t+1} = \theta_j^{t} + \alpha \sum_{i=1}^{n} (y_i - \vec{\mathbf{x}}_i^{T} \theta^t) x_i^j$$

- For a single training point, we have:

$$\Rightarrow \quad \theta^{t+1} = \theta^t + \alpha (y_i - \vec{\mathbf{x}}_i^{T} \theta^t) \vec{\mathbf{x}}_i$$

  - a "**stochastic**", "**coordinate**" descent algorithm
  - This can be used as an **on-line** algorithm

Method IV: Newton's method for optimization

- The most basic second-order optimization algorithm
- Updating parameter with

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{H}_K^{-1} \mathbf{g}_k$$

$$\Rightarrow \quad \theta^{t+1} = \theta^t - H^{-1} \nabla f(\theta)$$

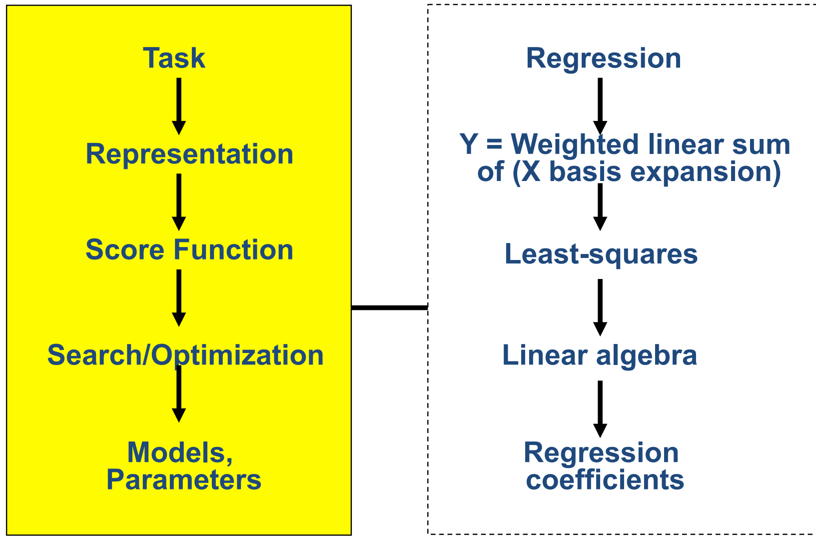
$$= \theta^t - (X^T X)^{-1} [X^T X \theta^t - X^T \vec{y}]$$

WHY ??? Normal Eq?

$$= (X^T X)^{-1} X^T y$$

Newton's method for Linear Regression

(2) Multivariate Linear Regression with basis Expansion



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

(2) LR with polynomial basis functions

- LR does not mean we can only deal with linear relationships

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

- E.g.: polynomial regression:

$$\varphi(x) := \left[1, x, x^2, x^3\right]$$

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

(3) Locally Weighted / Kernel Regression

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

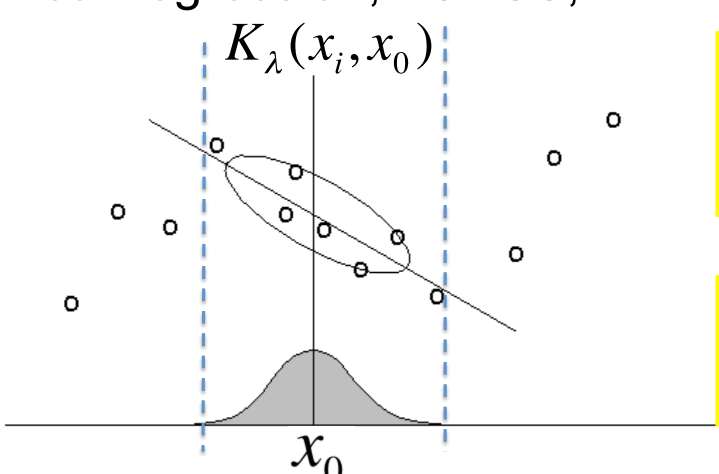
Regression → Y = local weighted linear sum of X's → Least-squares → Linear algebra → Local Regression coefficients (conditioned on each test point)

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_i,x_0)[y_i-\alpha(x_0)-\beta(x_0)x_i]^2$$

$$\hat{f}(x_0)=\hat{\alpha}(x_0)+\hat{\beta}(x_0)x_0$$

(3) Locally weighted regression

- *aka* locally weighted regression, locally linear regression, LOESS, …

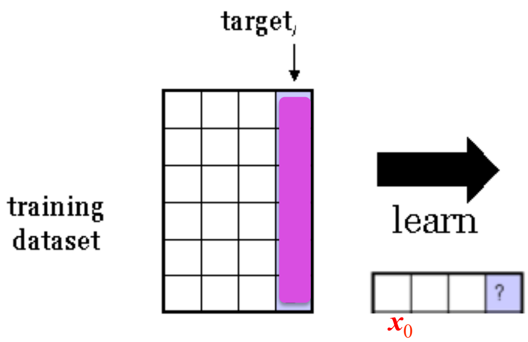


linear_func(x)->y ➔ could represent only the neighbor region of x_0

Use RBF function to pick out/emphasize the neighbor region of x_0 ➔ $K_\lambda(x_i,x_0)$

Figure 2: In locally weighted regression, points are weighted by proximity to the current x in question using a kernel. A regression is then computed using the weighted points.

LEARNING of Locally weighted linear regression



model

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$$

➔ Separate weighted least squares **at each target point x_0**

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_i, x_0)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2$$

(4) Regularized multivariate linear regression

Task	Regression
Representation	Y = Weighted linear sum of X's
Score Function	Least-squares
Search/Optimization	Linear algebra + Regularization
Models, Parameters	Regularized Regression coefficients

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

(4) LR with Regularizations / Regularized multivariate linear regression

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

41/54

Error on data + Regularization

What we have covered (II)

- Supervised Classification models
  - Support Vector Machine
  - Bayes Classifier
  - Logistic Regression
  - K-nearest Neighbor
  - Random forest / Decision Tree
  - Neural Network (e.g. MLP)
  - *Feature selection

Three major sections for classification

- We can divide the large variety of classification approaches into roughly three major types

1. Discriminative
   - directly estimate a decision rule/boundary
   - e.g., logistic regression, support vector machine, decisionTree
2. Generative:
   - build a generative statistical model
   - e.g., naïve bayes classifier, Bayesian networks
3. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K nearest neighbors

X_1	X_2	X_3	C

A Dataset for classification

$$f : \boxed{X} \longrightarrow \boxed{C}$$

Output as Discrete Class Label
$C_1, C_2, \ldots, C_L$

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

(1) Support Vector Machine

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

classification → Kernel Func K(xi, xj) → Margin -> Hinge Loss (optional) → QP with Dual form → Dual Weights

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

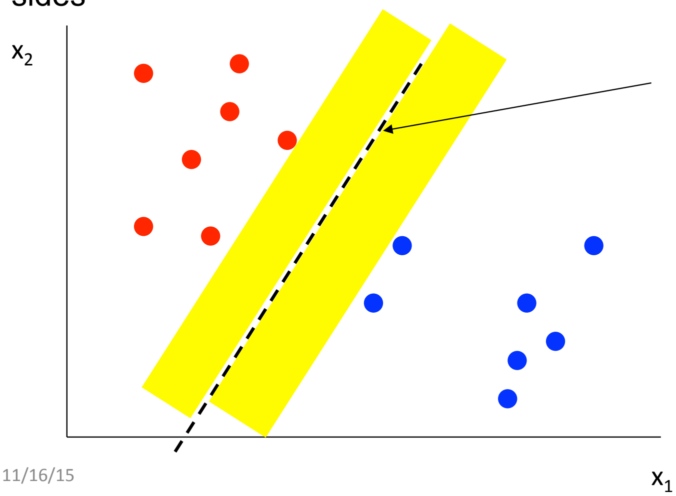
$$w = \sum_i \alpha_i x_i y_i$$

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

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

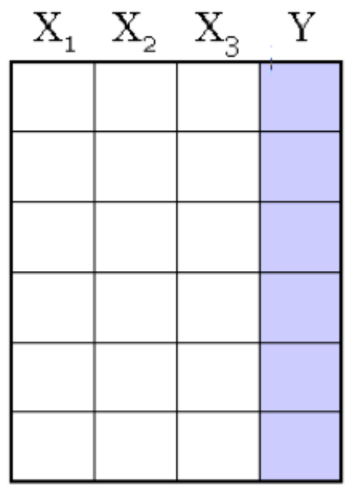
(1) SVM as Max margin classifiers

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



Why?

- Intuitive, 'makes sense'
- Some theoretical support
- Works well in practice



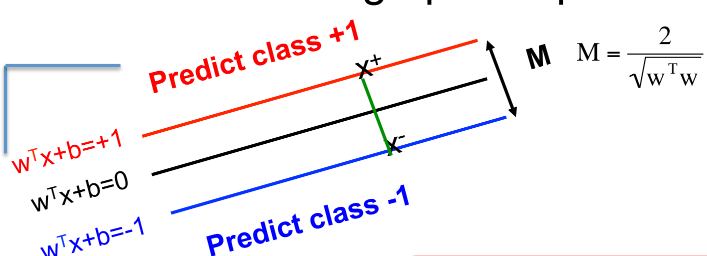
A Dataset for binary classification

$$f : X \longrightarrow Y$$

Output as Binary Class Label: 1 or -1

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

When linearly Separable Case: Optimization Step i.e. learning optimal parameter for SVM



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

$w^Tx+b <= -1$

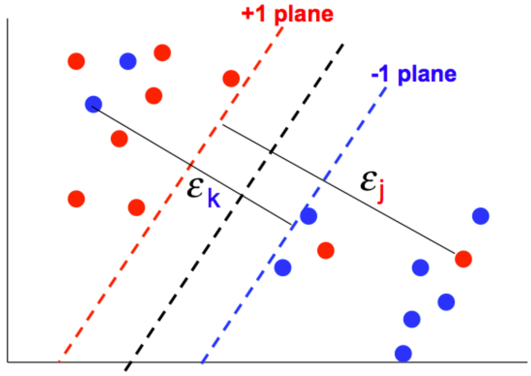
A total of n constraints if we have n input samples

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

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

SVM as a QP problem

Final optimization for non linearly separable case



The new optimization problem is:

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

subject to the following inequality constraints:

For all x_i in class + 1

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

For all x_i in class - 1

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

A total of n constraints

n

For all i

$$\varepsilon_i \geq 0$$

Another n constraints

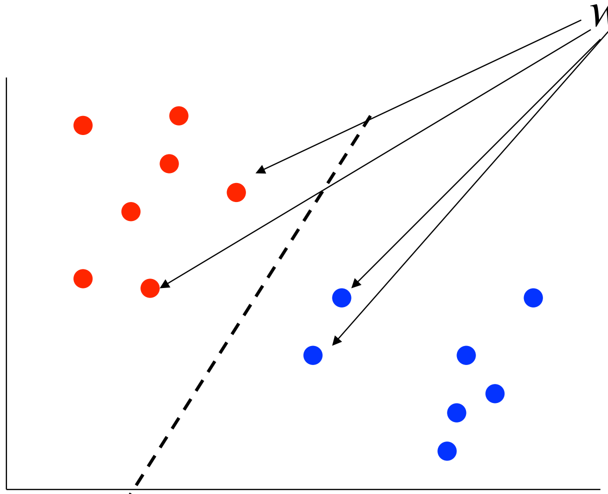
n

Dual SVM - interpretation

training points

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

$\alpha_i \geq 0$



For α_i that are 0, no influence

Dual SVM for linearly separable case

Our dual target function: $\max_\alpha \sum_{i=1}^{n_{train}} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ $\sum_{i=1}^{n_{train}} \sum_{j=1}^{n_{train}}$

Training

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dot product for all training samples

Dot product with training samples

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

Testing

$$w^T x_{ts} + b = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_{ts} + b$$

~~$\alpha_i \neq 0$~~ $\alpha_i > 0$

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

Dual formulation for linearly non separable case

Dual target function:

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

$$\sum_i \alpha_i y_i = 0$$

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

Hyperparameter C should be tuned through k-folds CV

The only difference is that the \alpha are now bounded

To evaluate a new sample x_j we need to compute:

$$w^T x_j + b = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_j + b$$

This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on a_i now

Once again, efficient algorithm exist to find a_i

The kernel trick

How many operations do we need for the dot product?

$$\Phi(x)^T\Phi(z) = \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_i \sum_{j=i+1} 2x_i x_j z_i z_j + 1$$

$O(m^2)$

m   m   m(m-1)/2   **=~ m²**

However, we can obtain dramatic savings by noting that

$$\Phi(x)^T\Phi(z) = (x^T z + 1)^2 = (x.z+1)^2 = (x.z)^2 + 2(x.z) + 1$$

$$= (\sum_i x_i z_i)^2 + \sum_i 2x_i z_i + 1$$

$$= \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_i \sum_{j=i+1} 2x_i x_j z_i z_j + 1$$

$K(x,z)$

We only need m operations!

$O(m)$

So, if we define the **kernel function** as follows, there is no need to carry out basis function \phi(.) explicitly

$$K(\mathbf{x},z) = (x^T z + 1)^2$$

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

(2) Bayes Classifier

Task	classification
↓	↓
Representation	Prob. models p(X|C)
↓	↓
Score Function	EPE with 0-1 loss, with Log likelihood(optional)
↓	↓
Search/Optimization	Many options
↓	↓
Models, Parameters	Prob. Models' Parameter

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

Gaussian Naive

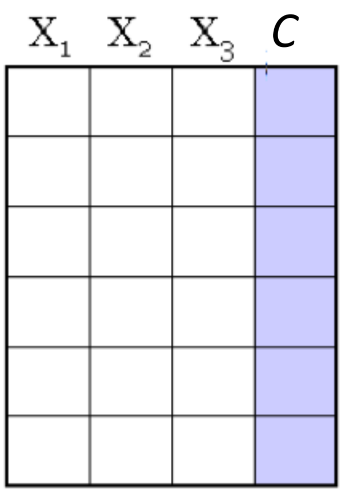
$$\hat{P}(X_j \mid C = c_k) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp\left(-\frac{(X_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right)$$

Bernoulli Naive

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

Multinomial

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



A Dataset for classification

$$f : X \longrightarrow C$$

Output as Discrete Class Label $C_1, C_2, \ldots, C_L$

Generative

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

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

Bayes classifier

- Treat each attribute and class label as random variables.
- Given a sample **x** with attributes ($x_1, x_2, \ldots, x_p$):
  - Goal is to predict class *C.*
  - Specifically, we want to find the value of C_i that maximizes $p(C_i \mid x_1, x_2, \ldots, x_p)$.
- Bayes classification

$$P(C \mid \mathbf{X}) \propto P(\mathbf{X} \mid C) P(C) = P(X_1, \cdots, X_p \mid C) P(C)$$

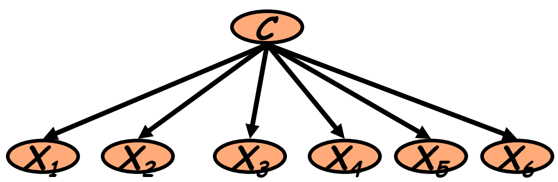
Difficulty: learning the joint probability $P(X_1, \cdots, X_p \mid C)$

Naïve Bayes Classifier

Difficulty: learning the joint probability $P(X_1, \cdots, X_p \mid C)$

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

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



Adapt from Prof. Ke Chen NB slides

Probabilistic Models of text documents

$w_1 .. w_K \in Voc$

$d_1 \vdots d_n$

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

$D = (w_1, w_2, \cdots, w_K)$

Two Previous models

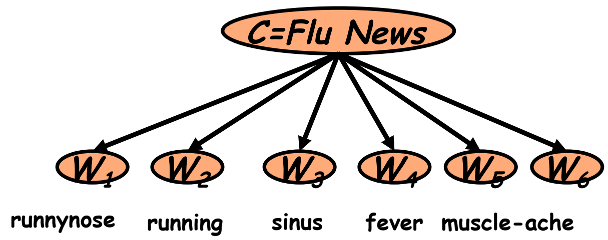
$$\Pr(W_1 = true, W_2 = false \ldots, W_k = true \mid C = c)$$

Multivariate Bernoulli Distribution

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

Multinomial Distribution

(2.1) : Multivariate Bernoulli for text



- **Conditional Independence Assumption:** Features (word presence) are *independent* of each other given the class variable:

this is naïve

Ber

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

- Multivariate Bernoulli model is appropriate for binary feature variables

(2.2) Multinomial Naïve Bayes as Stochastic Language Models

Model C1

0.2	the
0.01	boy
0.0001	said
0.0001	likes
0.0001	black
0.0005	dog
0.01	garden

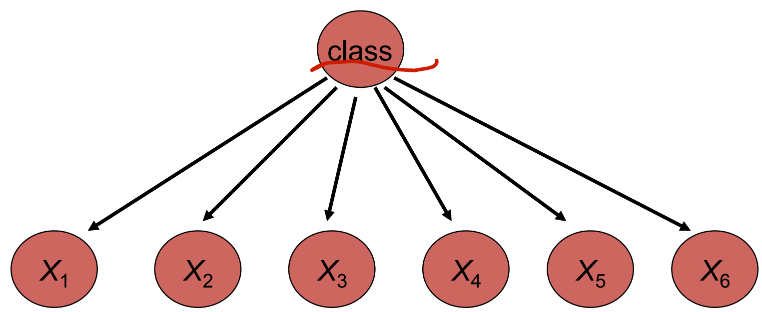
Model C2

0.2	the
0.0001	boy
0.03	said
0.02	likes
0.1	black
0.01	dog
0.0001	garden

the	boy	likes	black	dog
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

P(s|C2) P(C2) > P(s|C1) P(C1)

Multinomial Naïve Bayes = a class conditional unigram language model



- Think of X_i as the word on the i^{th} position in the document string
- Effectively, the probability of each class is done as a class-specific unigram language model

Adapt From Manning' textCat tutorial

(2.3) Gaussian Naïve Bayes Classifier

- Continuous-valued Input Attributes
  - Conditional probability modeled with the normal distribution

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

μ_{ji} : mean (avearage) of attribute values X_j of examples for which $C = c_i$

σ_{ji} : standard deviation of attribute values X_j of examples for which $C = c_i$

  - Learning Phase: for $\mathbf{X} = (X_1, \cdots, X_p)$, $C = c_1, \cdots, c_L$
    Output: $p \times L$ normal distributions and $P(C = c_i)$ $i = 1, \cdots, L$
  - Test Phase: for $\mathbf{X}' = (X'_1, \cdots, X'_p)$
    - Calculate conditional probabilities with all the normal distributions
    - Apply the MAP rule to make a decision

Naïve Gaussian means ?

$O(kp^2+kp)$

Σ_k

Not Naïve

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

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

p-dim random vector

Naïve

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

$$= \prod_i \frac{1}{\sqrt{2\pi}\sigma_{ji}}\exp\left(-\frac{(X_j-\mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

$O(kp+kp)$

Diagonal Matrix $\Sigma\_c_k = \Lambda\_c_k$

Each class' covariance matrix is diagonal

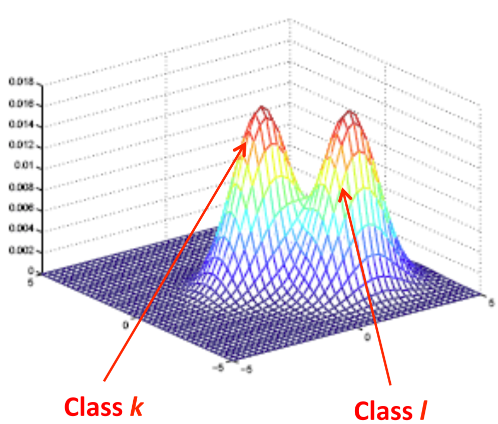
(2.4) LDA (Linear Discriminant Analysis)

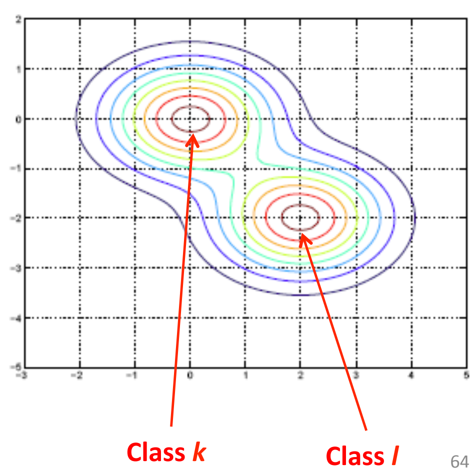
$O(p^2+kp)$

Linear Discriminant Analysis : $\Sigma_k = \Sigma, \ \forall k$

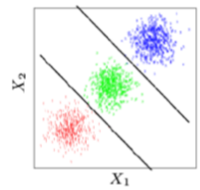
Each class' covariance matrix is the same

The Gaussian Distribution are shifted versions of each other





Optimal Classification



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

$$= \arg\max_k \Big[\cancel{-\log((2\pi)^{p/2}|\Sigma|^{1/2})} - \frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log(\pi_k) \Big]$$

$$= \arg\max_k \boxed{\Big[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log(\pi_k) \Big]}$$

Linear Discriminant Function for LDA

- Note

$$-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k - \frac{1}{2}x^T\Sigma^{-1}x$$

Define Linear Discriminant Function

$$\delta(x) = -\frac{1}{2}(x-\mu_k)^T \sum^{-1}(x-\mu_k) + log\pi_k$$

➔ The Decision Boundary Between class *k* and *l*, $\{x : \delta_k(x) = \delta_l(x)\}$, is a linear line/plane

$$\log\frac{P(C\_k \mid X)}{P(C\_l \mid X)} = \log\frac{P(X \mid C\_k)}{P(X \mid C\_l)} + \log\frac{P(C\_k)}{P(C\_l)}$$

$$= \log\frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k+\mu_\ell)^T\mathbf{\Sigma}^{-1}(\mu_k-\mu_\ell) + x^T\mathbf{\Sigma}^{-1}(\mu_k-\mu_\ell) \qquad (4.9)$$

Equals to zero

Boundary points X : when P(c_k|X) == P(c_l|X), the left linear equation ==0, a linear line / plane

(2.5) QDA (Quadratic Discriminant Analysis)

- Estimate the covariance matrix Σ_k separately for each class k, $k = 1, 2, ..., K$.
- *Quadratic discriminant function*:

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k \, .$$

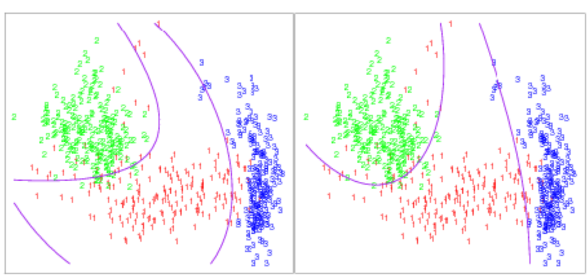
- Classification rule:

$$\hat{G}(x) = \arg\max_k \delta_k(x) \, .$$

- Decision boundaries are quadratic equations in x.
- QDA fits the data better than LDA, but has more parameters to estimate.

(2.6) LDA on Expanded Basis

- Expand input space to include X_1X_2, X_1^2, and X_2^2.
- Input is five dimensional: $X = (X_1, X_2, X_1X_2, X_1^2, X_2^2)$.



LDA with quadratic basis Versus QDA

Figure 4.6: *Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $x_1, x_2, x_{12}, x_1^2, x_2^2$). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is usually the case.*

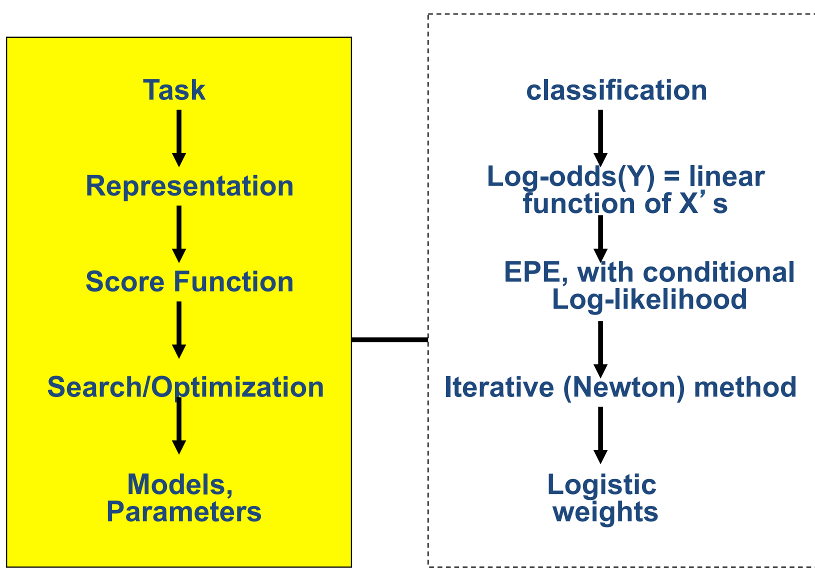
(2.7) Regularized Discriminant Analysis

- A compromise between LDA and QDA.
- Shrink the separate covariances of QDA toward a common covariance as in LDA.
- Regularized covariance matrices:

$$\hat{\Sigma}_k(\alpha) = \alpha\hat{\Sigma}_k + (1-\alpha)\hat{\Sigma} \, .$$

- The quadratic discriminant function $\delta_k(x)$ is defined using the shrunken covariance matrices $\hat{\Sigma}_k(\alpha)$.
- The parameter α controls the complexity of the model.

(3) Logistic Regression



$$P(c=1|x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$

Logistic Regression—when?

Logistic regression models are appropriate for target variable coded as 0/1.

Binary

We only observe "0" and "1" for the target variable—but we think of the target variable conceptually as a probability that "1" will occur.

p(y=1|x)

logistic

This means we use Bernoulli distribution to model the target variable with its Bernoulli parameter p=p(y=1|x) predefined.

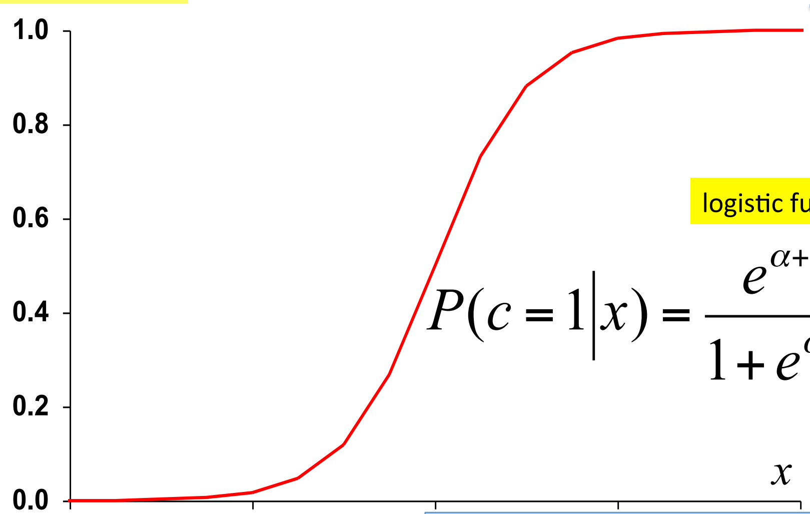
The main interest ➔ predicting the probability that an event occurs (i.e., the probability that p(y=1|x)).

Discriminative

Logistic regression models for binary target variable coded 0/1.

P (C=1|X)

e.g. Probability of disease



logistic function

$$P(c=1|x)=\frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$

Logit function

Decision Boundary ➔ equals to zero

$$\ln\left[\frac{P(c=1|x)}{P(c=0|x)}\right]=\ln\left[\frac{P(c=1|x)}{1-P(c=1|x)}\right]=\alpha+\beta_1x_1+\beta_2x_2+\ldots+\beta_px_p$$

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

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

(conditional) Log-likelihood:

How?

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

$p(y_i|x_i)$

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

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

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

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

Discriminative vs. Generative

Generative approach

- Model the joint distribution p(X, C) using

  p(X | C = c_k) and p(C = c_k)

Class prior

Discriminative approach

- Model the conditional distribution p(c| X) directly

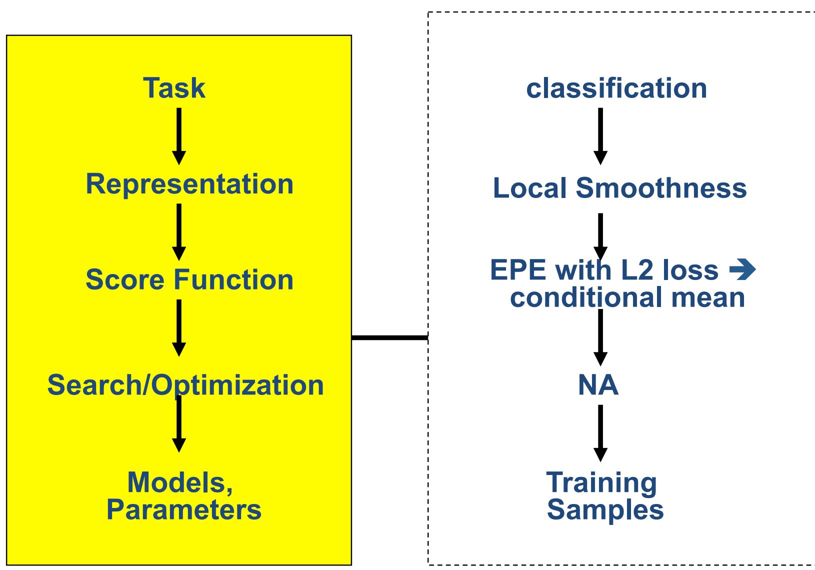
e.g.,

$$\frac{1}{1+e^{-(\beta_0+\beta_1 * X))}}$$

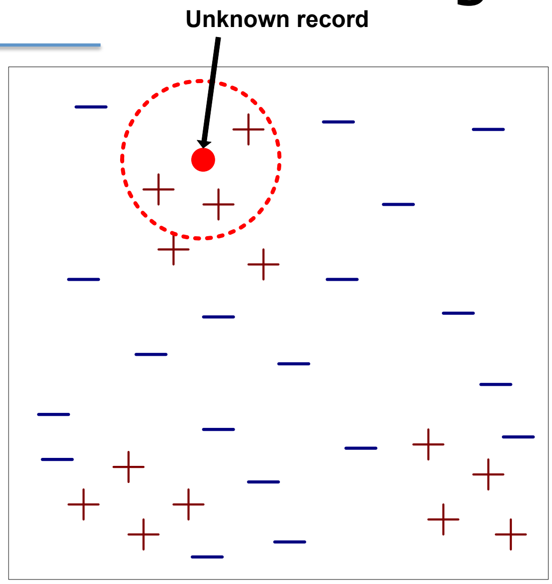
Discriminative vs. Generative

- Empirically, generative classifiers approach their asymptotic error faster than discriminative ones
  - Good for small training set
  - Handle missing data well (EM)
- Empirically, discriminative classifiers have lower asymptotic error than generative ones
  - Good for larger training set

(4) K-Nearest Neighbor



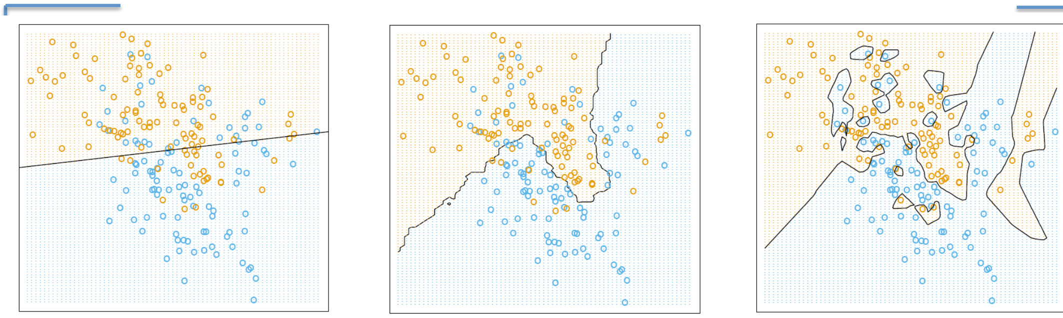
Nearest neighbor classifiers



To classify unknown sample:

1. Compute distance to other training records
2. Identify *k* nearest neighbors
3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Decision boundaries in global vs. local models



linear regression

- global
- stable
- can be inaccurate

15-nearest neighbor

1-nearest neighbor

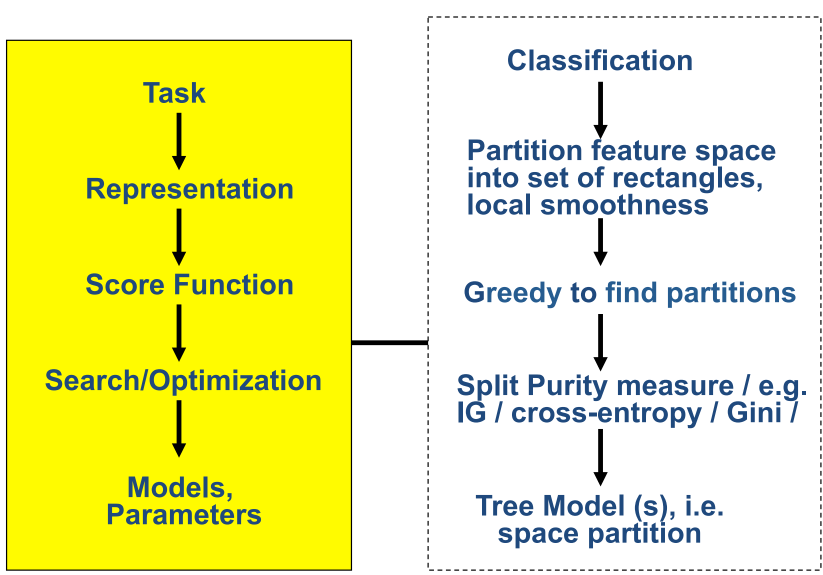
- local
- accurate
- unstable

What ultimately matters: ***GENERALIZATION***

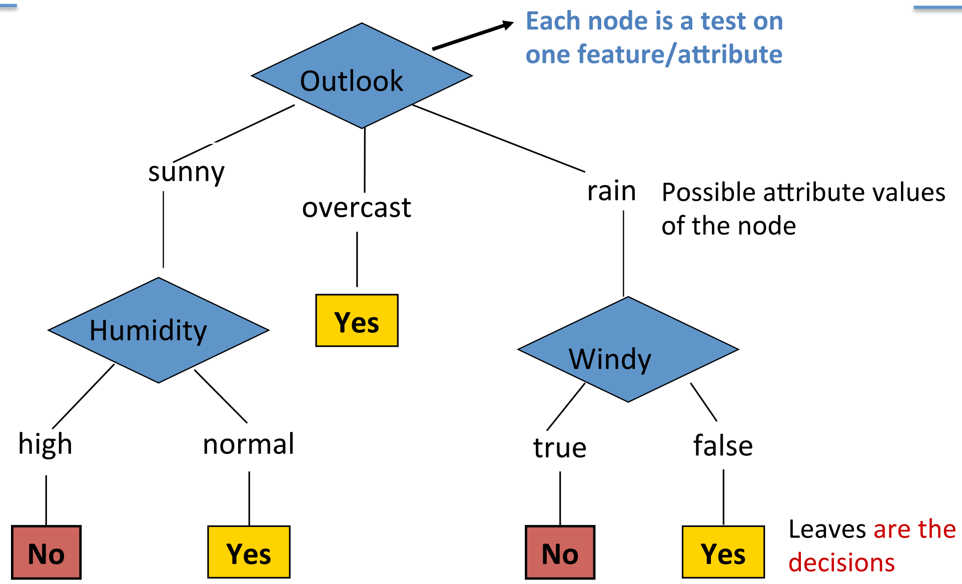
Nearest neighbor classification

- *k*-Nearest neighbor classifier is a lazy learner
  - Does not build model explicitly.
  - Unlike eager learners such as decision tree induction and rule-based systems.
  - Classifying unknown samples is relatively expensive.
- *k*-Nearest neighbor classifier is a local model, vs. global model of linear classifiers.

(5) Decision Tree / Random Forest



Anatomy of a decision tree



Decision trees

- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.

```
• (Outlook ==overcast)
•  OR
• ((Outlook==rain) and (Windy==false))
•  OR
• ((Outlook==sunny) and (Humidity=normal))
•  => yes play tennis
```

Information gain

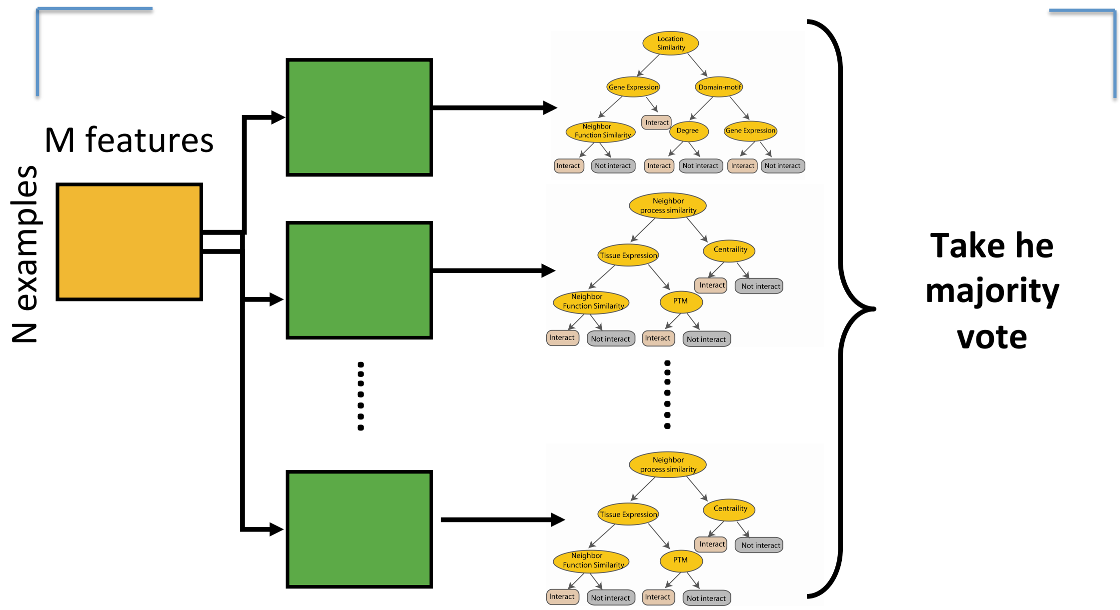
- IG(X_i,Y)=H(Y)-H(Y|X_i)

Reduction in uncertainty by knowing a feature X_i

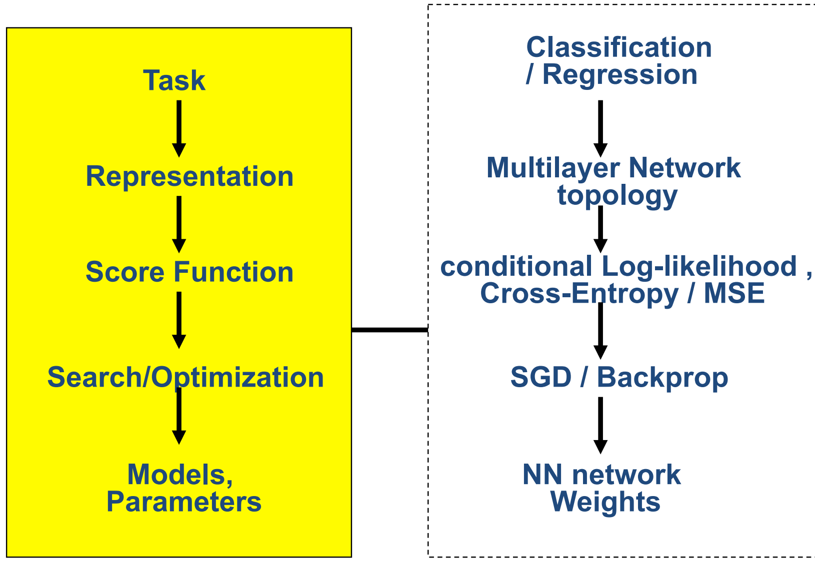
Information gain:
= (information before split) – (information after split)
= entropy(parent) – [average entropy(children)]

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

Random Forest Classifier



(6) Neural Network



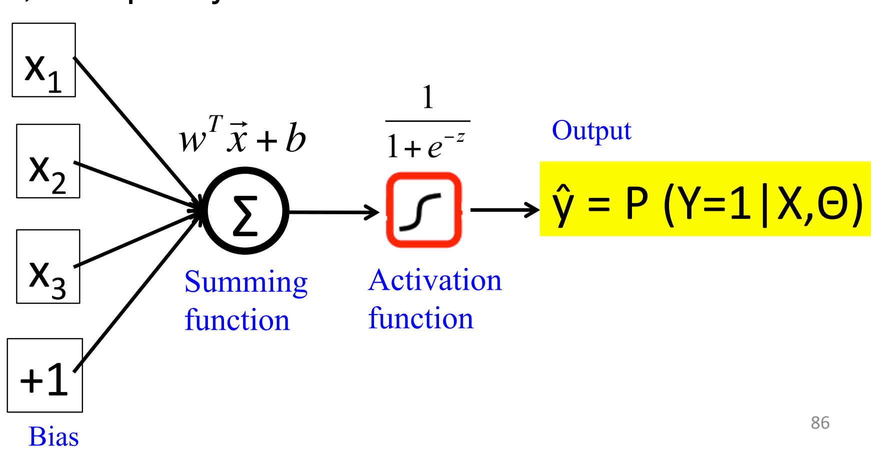
Logistic regression

Logistic regression could be illustrated as a module

On input x, it outputs ŷ:

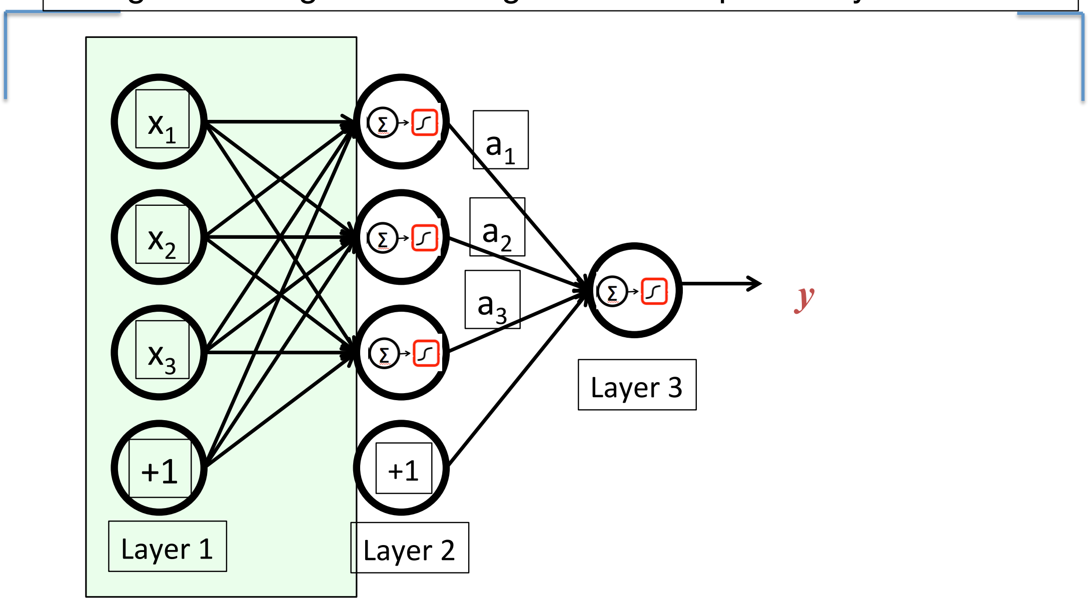
where

Draw a logistic regression unit as:

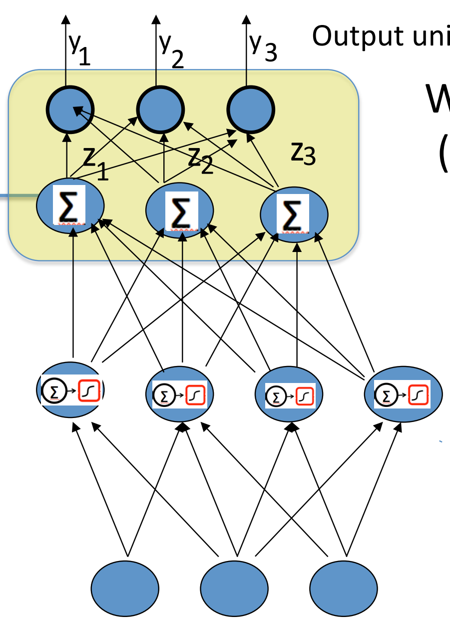


Multi-Layer Perceptron (MLP)

String a lot of logistic units together. Example: 3 layer network:

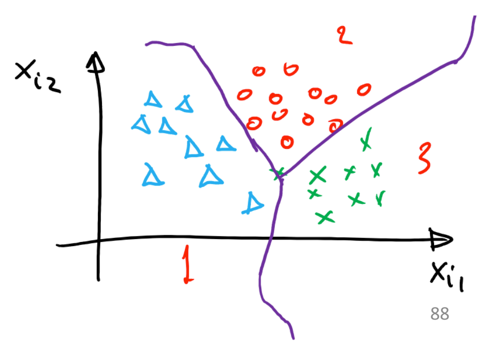


When for multi-class classification (last output layer: softmax layer)



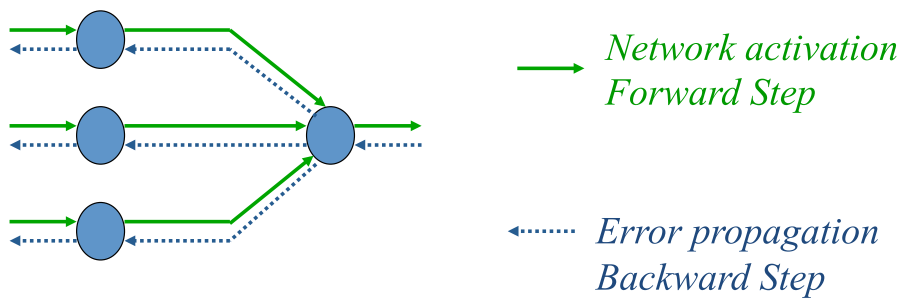
When multi-class output, last layer is softmax output layer ➔ a multinomial logistic regression unit

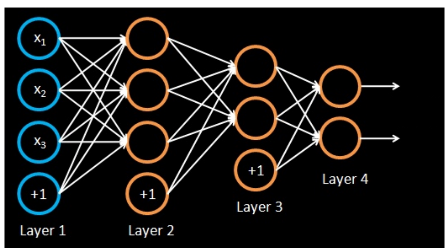
Y_{i1}	Y_{i2}	Y_{i3}	
0	1	0	Class 2
1	0	0	Class 1
0	0	1	Class 3



Backpropagation

- Back-propagation training algorithm





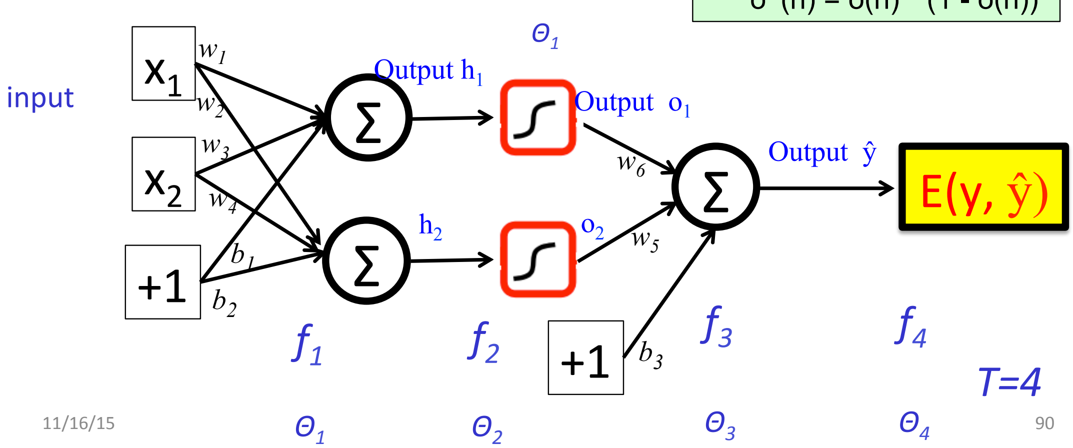
to train this layered network. The stacked layers in our network can be written in a more general form of multi-level functions:

$$l_{\mathbf{x}} = \mathbf{f}_T(\mathbf{f}_{T-1}(...(\mathbf{f}_1(\mathbf{x}))...)),$$

where $l_{\mathbf{x}}$ denotes the loss on a single example $\mathbf{x}$

For instance → for regression

for sigmoid unit o, its derivative is,
o'(h) = o(h) * (1 - o(h))



$\mathbf{f}_i, i \in [1, T]$, the derivative for updating its parameter set $\boldsymbol{\theta}_i$ is using the delta rule:

$$\frac{\partial l}{\partial \boldsymbol{\theta}_i} = \boxed{\frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_i}} \times \frac{\partial \mathbf{f}_i}{\partial \boldsymbol{\theta}_i},$$

and the first factor on the right can be recursively calculated:

$$\boxed{\frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_i}} = \frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_{i+1}} \times \frac{\partial \mathbf{f}_{i+1}}{\partial \mathbf{f}_i}.$$

Note that $\mathbf{f}$ and θ are usually vectors so $\frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_{i+1}}$ and $\frac{\partial \mathbf{f}_i}{\partial \boldsymbol{\theta}_i}$ are Jacobian matrices, and "$\times$" is matrix multiplication.

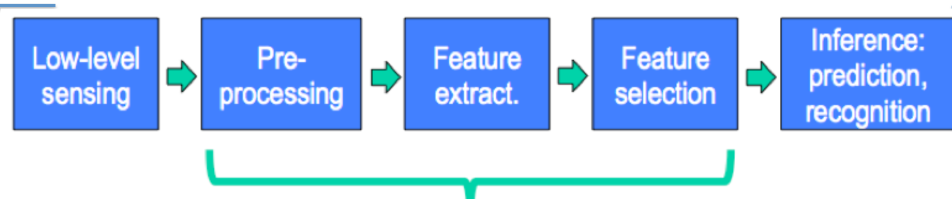
e.g.

$$\frac{\partial f_4}{\partial f_3} = \frac{\partial (y - \underbrace{\hat{y}}_{f_3(f_2(f_1(x)))})^2}{\partial f_3} = -2\underbrace{(y - \hat{y})}_{\text{output error}}$$

for $j = 1$ to MaxIter **do**
  if converge **then**
    break
  end if
  $\mathbf{x}, y \leftarrow$ random sampled data point and label
  calculate loss $l(\mathbf{x}; y)$
  cumulative $\leftarrow 1$
  for $i = \mathrm{T}$ to 1 **do**
    $\frac{\partial l}{\partial \boldsymbol{\theta}_i} \leftarrow \boxed{\text{cumulative}} * \frac{\partial \mathbf{f}_i}{\partial \boldsymbol{\theta}_i}$
    $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \lambda \frac{\partial l}{\partial \boldsymbol{\theta}_i}$
    $\boxed{\text{cumulative}} \leftarrow \boxed{\text{cumulative}} * \frac{\partial \mathbf{f}_{i+1}}{\partial \mathbf{f}_i}$
  end for
end for

Error propagation Backward Step

Deep Learning Way:
Learning features / Representation from data



Feature Engineering
- ✓ Most critical for accuracy
- ✓ Account for most of the computation for testing
- ✓ Most time-consuming in development cycle
- ✓ Often hand-craft and task dependent in practice

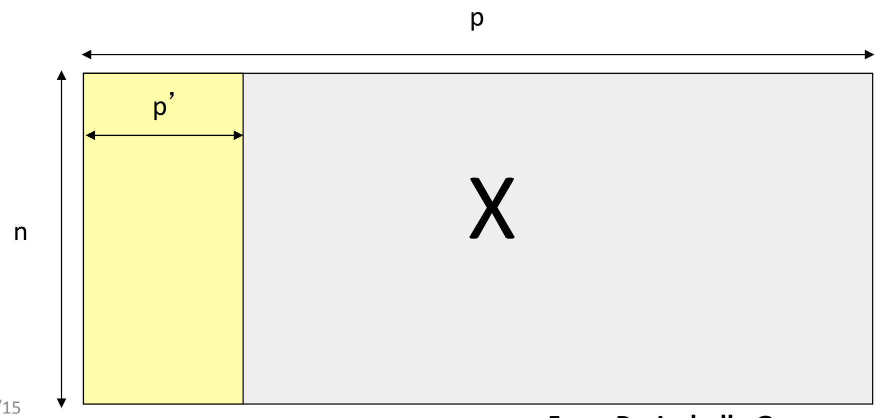
Feature Learning
- ✓ Easily adaptable to new similar tasks
- ✓ Layerwise representation
- ✓ Layer-by-layer unsupervised training
- ✓ Layer-by-layer supervised training

DESIGN ISSUES for **Deep NN**

- Data representation
- Network Topology
- Network Parameters
- Training
  - Scaling up with **graphics processors**
  - Scaling up with **Asynchronous SGD**
- Validation

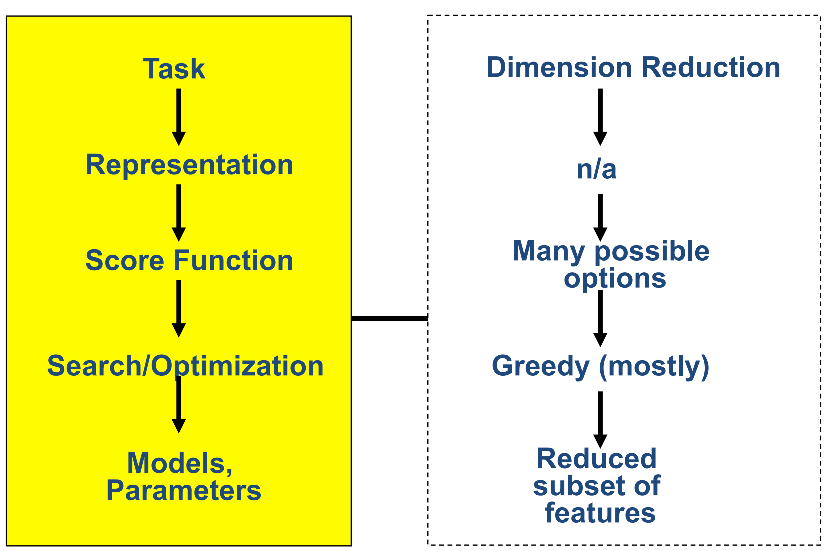
(7) Feature Selection

- **Thousands to millions of low level features**: select the most relevant one to build **better, faster, and easier to understand** learning machines.



From Dr. **Isabelle Guyon**

(7) Feature Selection (not covered)



Feature Selection

– Filtering approach:
ranks features or feature subsets independently of the predictor (classifier).
  - ...using univariate methods: consider one variable at a time
  - ...using multivariate methods: consider more than one variables at a time

– Wrapper approach:
uses a classifier to assess (many) features or feature subsets.

– Embedding approach:
uses a classifier to build a (single) model with a subset of features that are internally selected.

What we have covered (III)

- Unsupervised models
  – Dimension Reduction (PCA)
  – Hierarchical clustering
  – K-means clustering
  – GMM/EM clustering

An unlabeled Dataset X

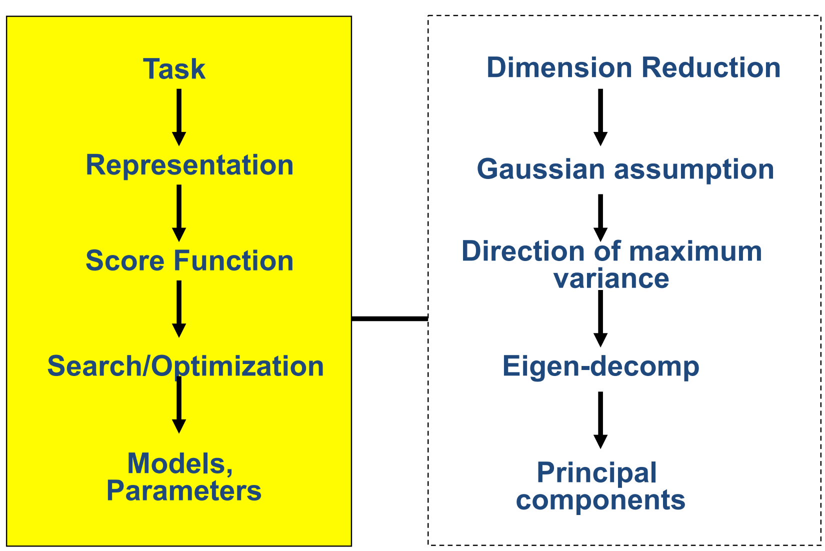
	X_1	X_2	X_3
s_1			
s_2			
s_3			
s_4			
s_5			
s_6			

a data matrix of n observations on p variables $x_1, x_2, \dots x_p$

Unsupervised learning = learning from raw (unlabeled, unannotated, etc) data, as opposed to supervised data where a label of examples is given

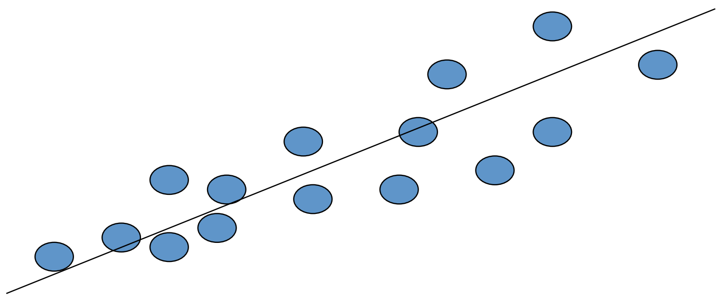
- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns]

(0) Principal Component Analysis



Algebraic Interpretation – 1D

- Given n points in a p dimensional space, for large p, how does one project on to a 1 dimensional space?



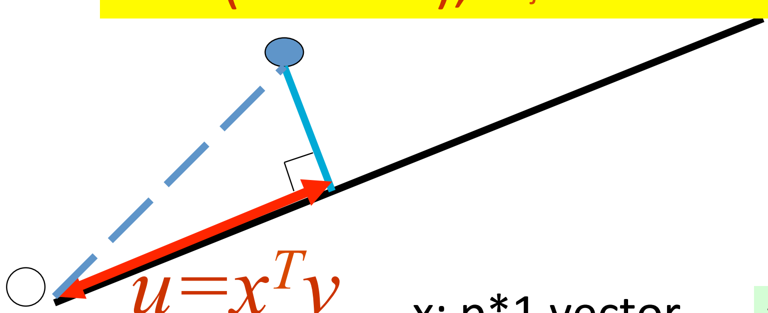
- Choose a line that fits the data so the points are spread out well along the line

From Dr. S. Narasimhan

Algebraic Interpretation – 1D

- Minimizing sum of squares of distances to the line is the same as maximizing the sum of squares of the projections on that line, thanks to Pythagoras.

$$\max(v^T X^T Xv), \text{ subject to } v^T v = 1$$

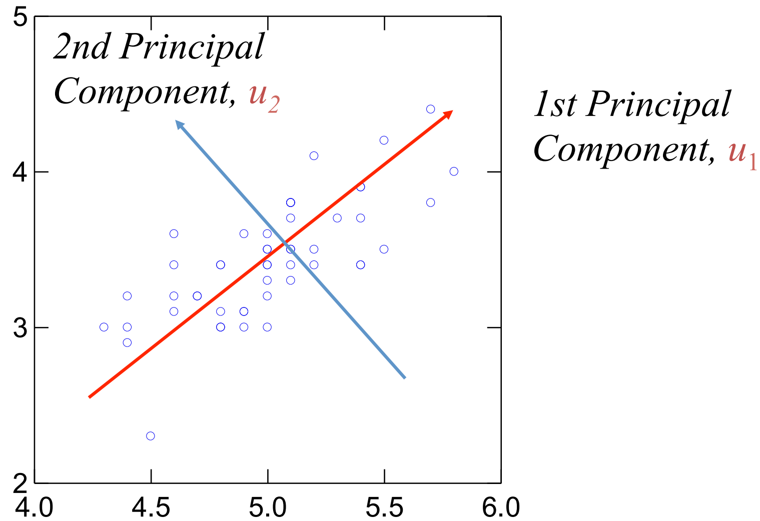


x: p*1 vector

v: p*1 vector

assuming data is centered

PCA Eigenvectors ➔ Principal Components



PCA & Gaussian Distributions.

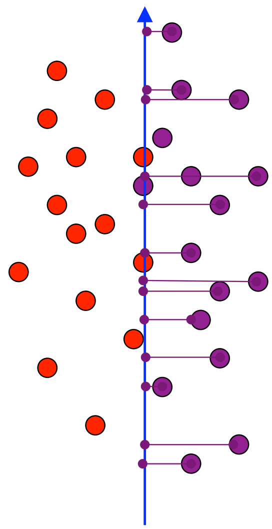
- PCA is similar to learning a Gaussian distribution for the data.

- Dimension reduction occurs by ignoring the directions in which the covariance is small.

PCA Limitations

- The direction of maximum variance is not always good for classification

not ideal for discrimination



First PC

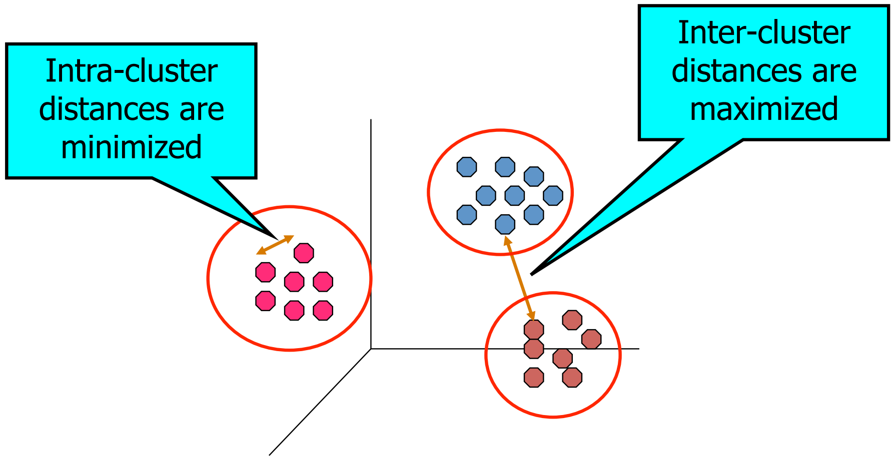
From Prof. Derek Hoiem

What we have covered (III)

- ❑ Unsupervised models
  - Dimension Reduction (PCA)
  - Hierarchical clustering
  - K-means clustering
  - GMM/EM clustering

What is clustering?

- Find groups (clusters) of data points such that data points in a group will be similar (or related) to one another and different from (or unrelated to) the data points in other groups

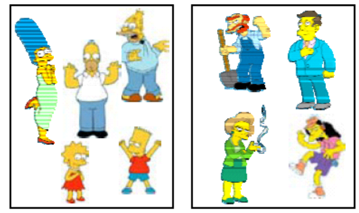


Issues for clustering

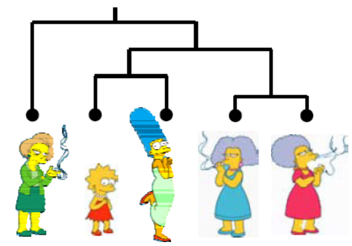
- What is a natural grouping among these objects?
  - Definition of "groupness"
- What makes objects "related"?
  - Definition of "similarity/distance"
- Representation for objects
  - Vector space? Normalization?
- How many clusters?
  - Fixed a priori?
  - Completely data driven?
    - Avoid "trivial" clusters - too large or small
- Clustering Algorithms
  - Partitional algorithms
  - Hierarchical algorithms
- Formal foundation and convergence

Clustering Algorithms

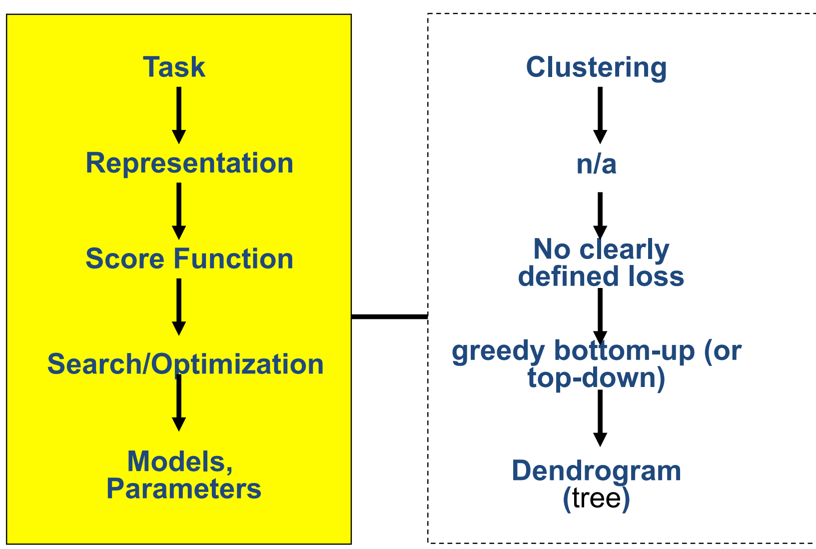
- Partitional algorithms
  - Usually start with a random (partial) partitioning
  - Refine it iteratively
    - K means clustering
    - Mixture-Model based clustering



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



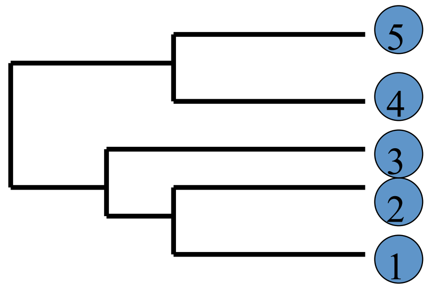
(1) Hierarchical Clustering



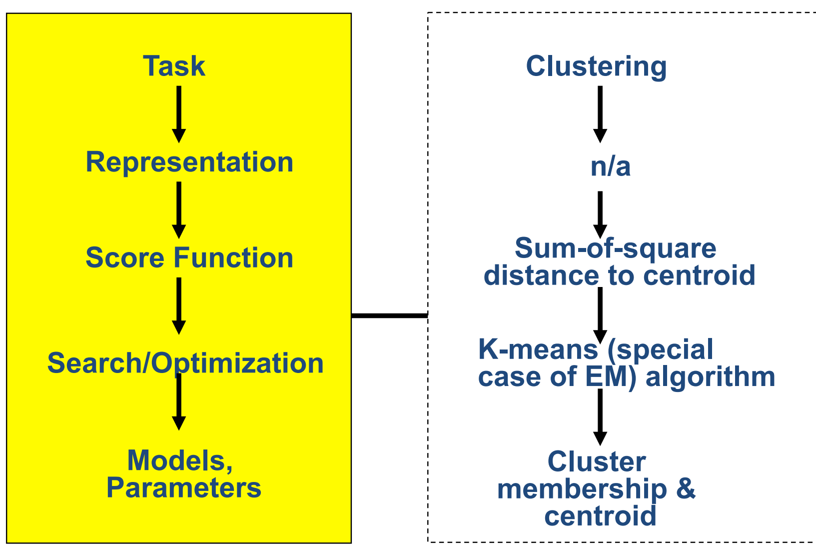
Example: single link

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & & & & \\ 2 & 2 & 0 & & & \\ 3 & 6 & 3 & 0 & & \\ 4 & 10 & 9 & 7 & 0 & \\ 5 & 9 & 8 & 5 & 4 & 0 \end{array} \Rightarrow \begin{array}{c|cccc} & (1,2) & 3 & 4 & 5 \\ \hline (1,2) & 0 & & & \\ 3 & 3 & 0 & & \\ 4 & 9 & 7 & 0 & \\ 5 & 8 & 5 & 4 & 0 \end{array} \Rightarrow \begin{array}{c|ccc} & (1,2,3) & 4 & 5 \\ \hline (1,2,3) & 0 & & \\ 4 & 7 & 0 & \\ 5 & 5 & 4 & 0 \end{array}$$

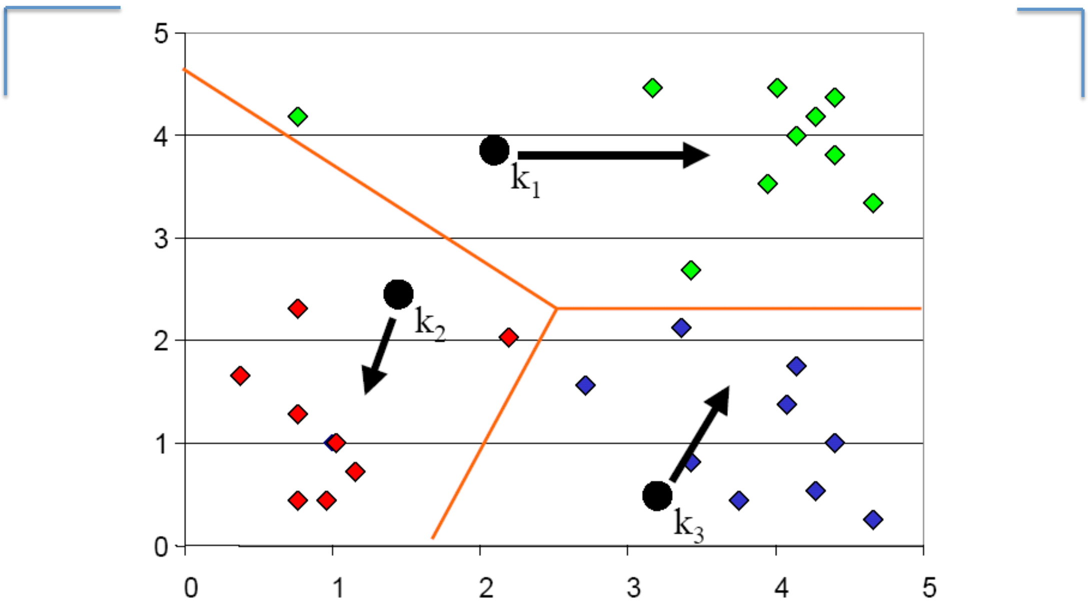
$$d_{(1,2,3),(4,5)} = \min\{ d_{(1,2,3),4}, d_{(1,2,3),5} \} = 5$$



(2) K-means Clustering



K-means Clustering: Step 2
- Determine the membership of each data points



(3) GMM Clustering

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

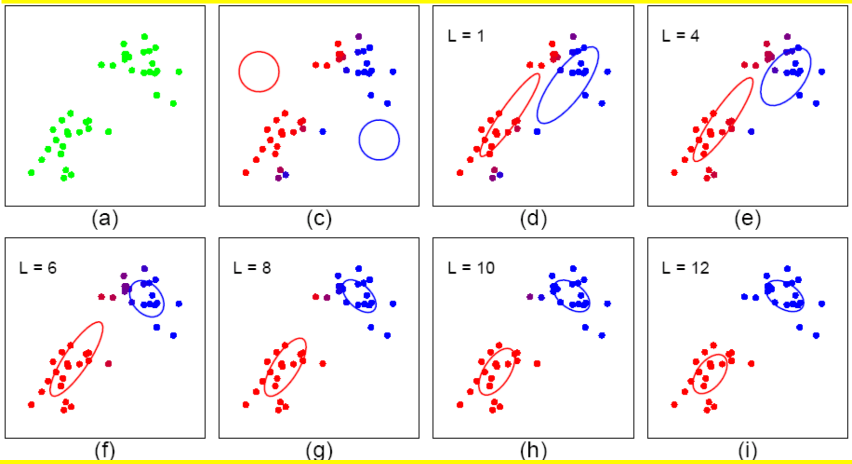
Clustering → **Mixture of Gaussians** → **Likelihood** → **EM algorithm** → **Each point's soft membership & mean / covariance per cluster**

$$\sum_i \log p(x = x_i) = \sum_i \log \left[\sum_{\mu_j} p(\mu = \mu_j) \frac{1}{\left(2\pi\sigma^2\right)^{d/2}} \exp\left(-\frac{\left\| x_i - \mu_j \right\|_2}{2\sigma^2} \right) \right]$$

Expectation-Maximization for training GMM

- Start:
  - "Guess" the centroid m_k and covariance S_k of each of the K clusters
- Loop

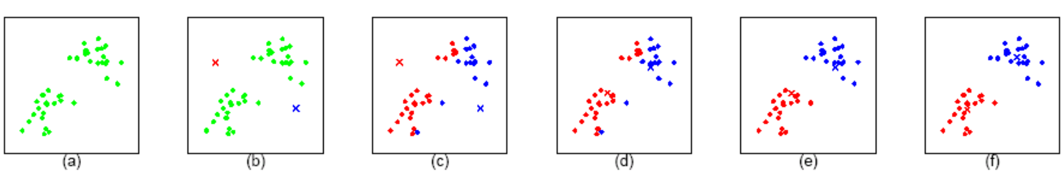
each cluster, revising both the mean (centroid position) and covariance (shape)



For each point, revising its proportions belonging to each of the K clusters

Compare: K-means

- The EM algorithm for mixtures of Gaussians is like a "soft version" of the K-means algorithm.
- In the K-means "E-step" we do hard assignment:
- In the K-means "M-step" we update the means as the weighted sum of the data, but now the weights are 0 or 1:



What we have covered (IV)

- ❑ Learning theory / Model selection
  - – K-folds cross validation
  - – Expected prediction error
  - – Bias and variance tradeoff

Dr. Yanjun Qi / UVA CS 6316 / f15

(1) Evaluation Choice:

e.g. 10 fold Cross Validation

- Divide data into 10 equal pieces
- 9 pieces as training set, the rest 1 as test set
- Collect the scores from the diagonal

model	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	train	train	train	train	train	train	train	train	train	test
2	train	train	train	train	train	train	train	train	test	train
3	train	train	train	train	train	train	train	test	train	train
4	train	train	train	train	train	train	test	train	train	train
5	train	train	train	train	train	test	train	train	train	train
6	train	train	train	train	test	train	train	train	train	train
7	train	train	train	test	train	train	train	train	train	train
8	train	train	test	train	train	train	train	train	train	train
9	train	test	train	train	train	train	train	train	train	train
10	test	train	train	train	train	train	train	train	train	train

CV-based Model Selection
We're trying to decide which algorithm to use.

- We train each model and make a table...

i	f_i	TRAINERR	10-FOLD-CV-ERR	Choice
1	f_1			
2	f_2			
3	f_3			√
4	f_4			
5	f_5			
6	f_6			

Which kind of cross-validation ?

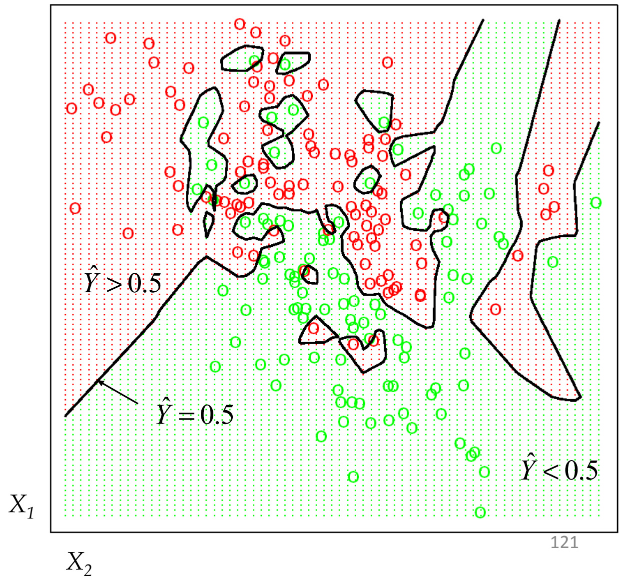
	Downside	Upside
Test-set	Variance: unreliable estimate of future performance	Cheap
Leave-one-out	Expensive. Has some weird behavior	Doesn't waste data
10-fold	Wastes 10% of the data. 10 times more expensive than test set	Only wastes 10%. Only 10 times more expensive instead of R times.
3-fold	Wastier than 10-fold. Expensivier than test set	Slightly better than test-set
R-fold	Identical to Leave-one-out	

(2) e.g. Training Error from KNN, Lesson Learned

- When k = 1,
- No misclassifications (on training): Overtraining

- Minimizing training error is not always good (e.g., 1-NN)

1-nearest neighbor averaging



Statistical Decision Theory

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

- Expected prediction error (EPE):
- $$\mathrm{EPE}(f) = \mathrm{E}(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$
  $$\text{e.g.} = \int (y - f(x))^2 \Pr(dx, dy)$$

Consider population distribution

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

Expected prediction error (EPE)

Consider joint distribution

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

- For L2 loss: e.g. $= \int (y - f(x))^2 \Pr(dx, dy)$

under L2 loss, best estimator for EPE (Theoretically) is :

Conditional mean $\hat{f}(x) = \mathrm{E}(Y \mid X = x)$

ts

e.g. KNN

NN methods are the direct implementation (approximation)

- For 0-1 loss: $L(k, \ell) = 1 - d_{kl}$

$\hat{f}(X) = C_k$ if

$\Pr(C_k \mid X = x) = \max_{g \in C} \Pr(g \mid X = x)$

Bayes Classifier

(3) Bias-Variance Trade-off

E(MSE) = noise^2 + bias^2 + variance

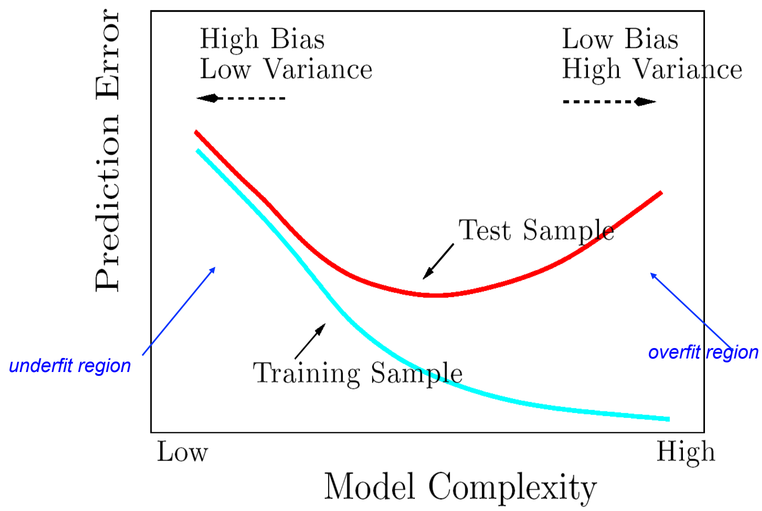
Unavoidable error

Error due to incorrect assumptions

Error due to variance of training samples

Slide credit: D. Hoiem

Bias-Variance Tradeoff / Model Selection



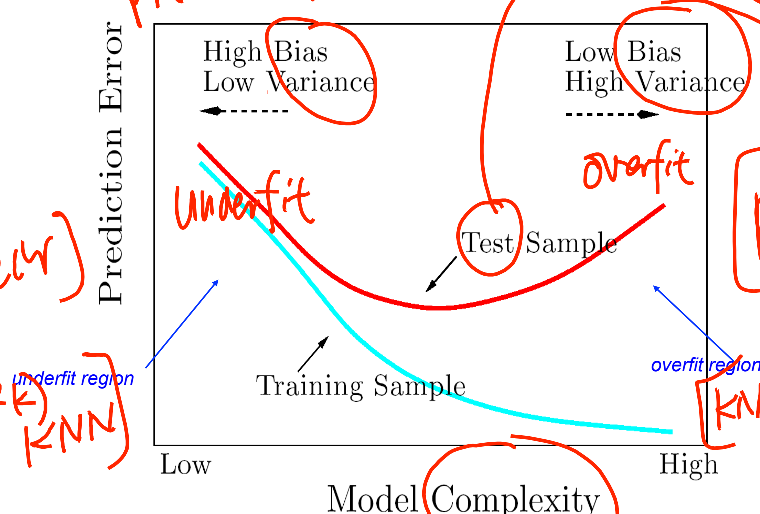
Bias-Variance Tradeoff / Model Selection

$Bias^2 + Vars$ → test error // k-folds CV error ⇓ good approximation of EPE

Pr(x,y)

[linear]

[(large k) KNN]

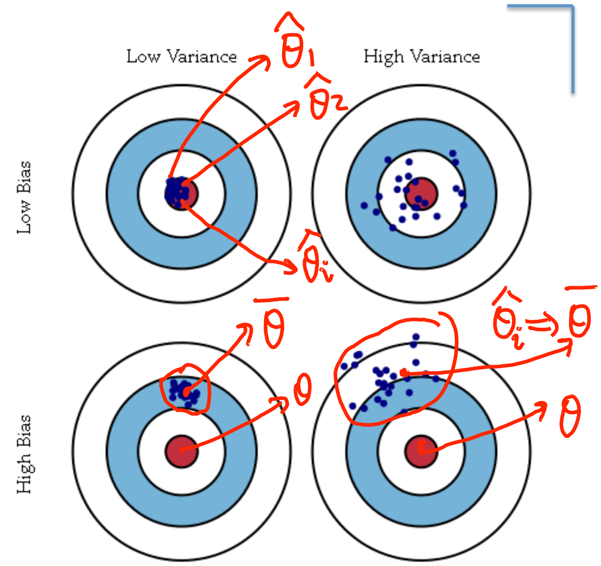


[polynomial p large]

[KNN (K=1)]

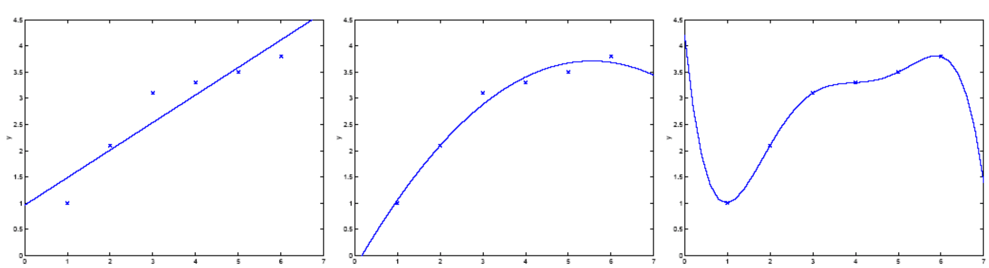
Model "bias" & Model "variance"

- Middle RED:
  - TRUE function
- Error due to bias:
  - How far off in general from the middle red
- Error due to variance:
  - How wildly the blue points spread



$E(\theta - \bar{\theta})$

Which function *f to choose*? Many possible choices , e.g. LR with polynomial basis functions

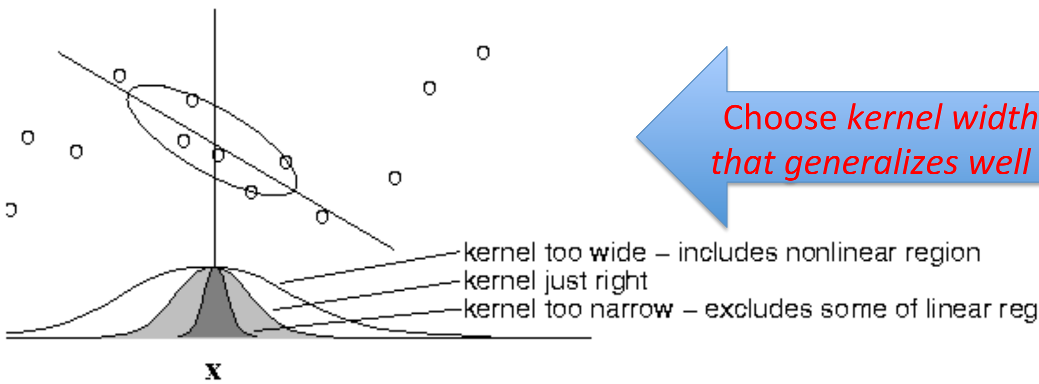


$y = \theta_0 + \theta_1 x$ $\qquad$ $y = \theta_0 + \theta_1 x + \theta_2 x^2$ $\qquad$ $y = \sum_{j=0}^{5} \theta_j x^j$

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

Choose *f that generalizes well !*

Which kernel width to choose ? e.g. locally weighted LR



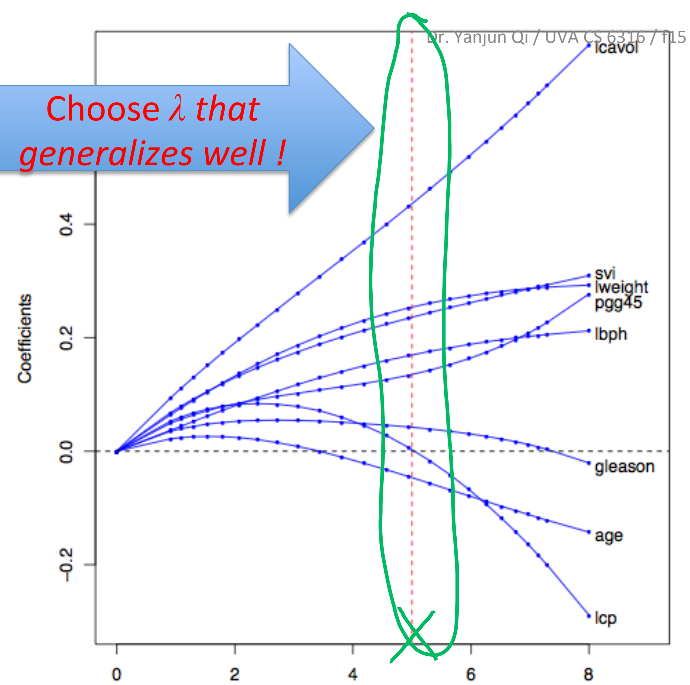
Choose *kernel width that generalizes well !*

Figure 3: The estimator variance is minimized when the kernel includes as many training points as can be accommodated by the model. Here the linear LOESS model is shown. Too large a kernel includes points that degrade the fit; too small a kernel neglects points that increase confidence in the fit.

Which regularization parameter to choose ?

Ridge Regression

Choose λ *that generalizes well !*



$\lambda \rightarrow \infty$ $\qquad$ $\lambda = 0$

λ increases

when varying λ, *how θ_j varies.*

Choose λ *that generalizes well !*

Regularization path of a Lasso Estimator

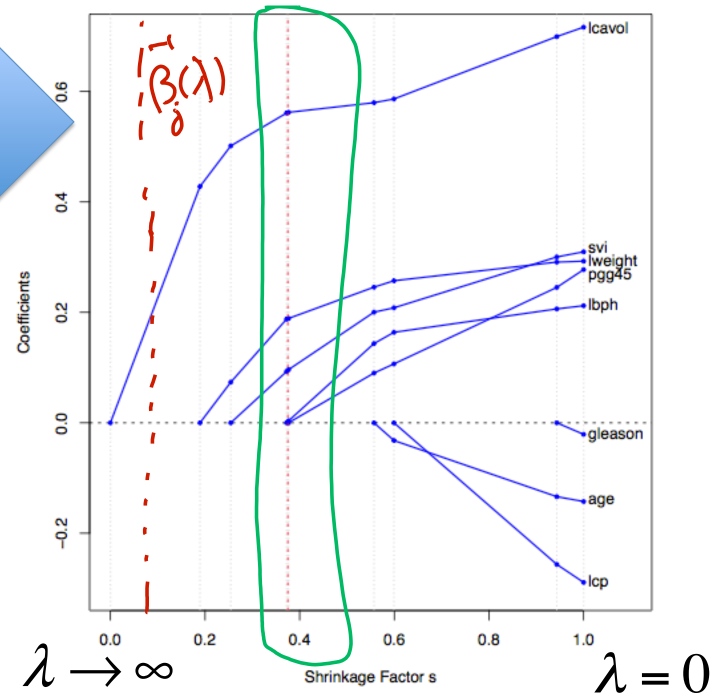


FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus $s = t/\sum_1^p |\hat{\beta}_j|$. A vertical line is drawn at $s = 0.36$, the value chosen by cross-validation. Compare Figure 3.8 on page 65; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed; see Section 3.4.4 for details.*

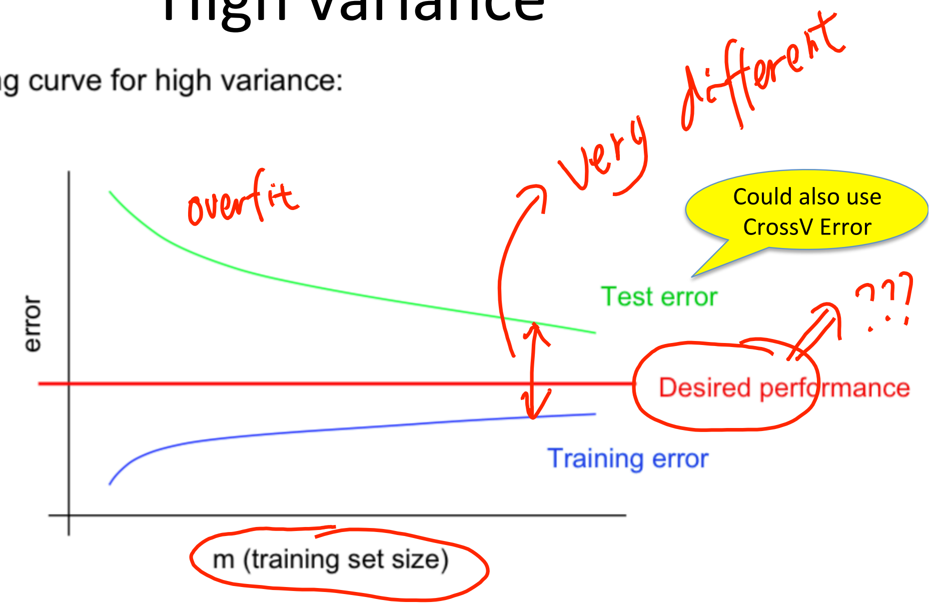
need to make assumptions that are able to generalize

- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model is too "simple" to represent all the relevant class 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

High variance

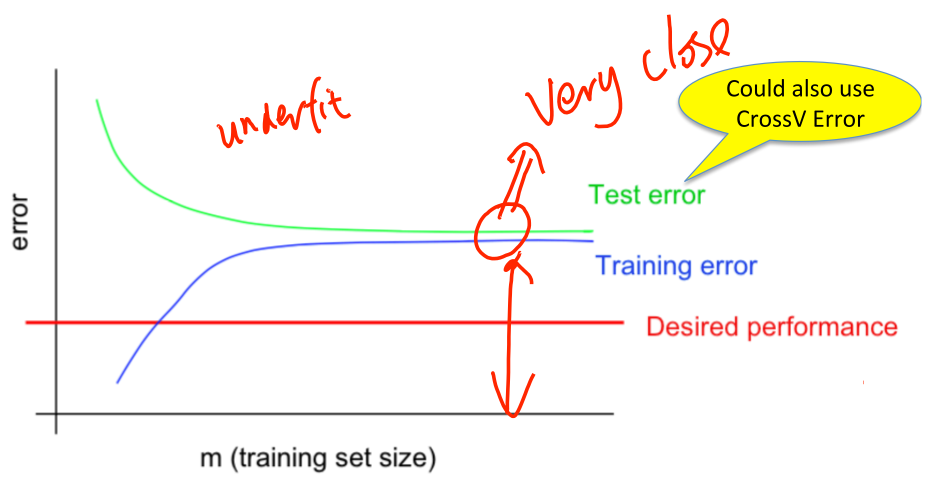
Typical learning curve for high variance:



- Test error still decreasing as m increases. Suggests larger training set will help.
- Large gap between training and test error.
- Low training error and high test error

High bias

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

What we have covered for each component

Task	Regression, classification, clustering, dimen-reduction
Representation	Linear func, nonlinear function (e.g. polynomial expansion), local linear, logistic function (e.g. p(c|x)), tree, multi-layer, prob-density family (e.g. Bernoulli, multinomial, Gaussian, mixture of Gaussians), local func smoothness, kernel matrix
Score Function	MSE, Hinge (margin), log-likelihood, EPE (e.g. L2 loss for KNN, 0-1 loss for Bayes classifier), cross-entropy, cluster points distance to centers, variance,
Search/ Optimization	Normal equation, gradient descent, stochastic GD, Newton, Linear programming, Quadratic programming (quadratic objective with linear constraints), greedy, EM, asyn-SGD, eigenDecomp
Models, Parameters	Regularization (e.g. L1, L2)

References

❑ Hastie, Trevor, et al. The elements of statistical learning. Vol. 2. No. 1. New York: Springer, 2009.

❑ Prof. M.A. Papalaskar's slides

❑ Prof. Andrew Ng's slides