

Decision tree, kNN and model selection

Yifeng Tao

School of Computer Science
Carnegie Mellon University

Slides adapted from Ziv Bar-Joseph, Tom Mitchell, Eric Xing

Outline

- Decision tree and random forest
- kNN
- Model selection and overfitting

Decision trees

- One of the most intuitive classifiers
- Easy to understand and construct
- Surprisingly, also works very well

The 'best' classifier

Gradient boosted machines and deep neural nets have dominated recent Kaggle competitions

Competition	Type	Winning ML Library/Algorithm
Liberty Mutual	Regression	XGBoost
Caterpillar Tubes	Regression	Keras + XGBoost + Reg. Forest
Diabetic Retinopathy	Image	SparseConvNet + RF
Avito	CTR	XGBoost
Taxi Trajectory 2	Geostats	Classic neural net
Grasp and Lift	EEG	Keras + XGBoost + other CNN
Otto Group	Classification	Stacked ensemble of 35 models
Facebook IV	Classification	sklearn GBM

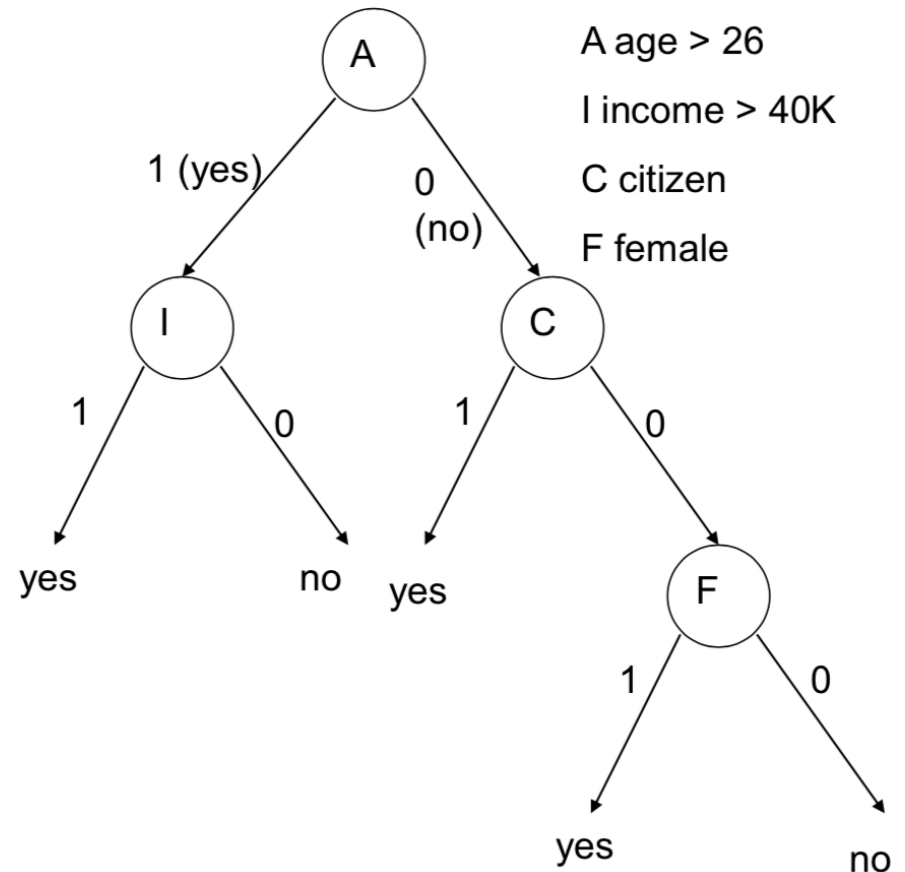
- They are quite robust, intuitive and, surprisingly, very accurate
- XGBoost: an ensemble method of decision trees

[Slide from <https://www.quora.com/What-machine-learning-approaches-have-won-most-Kaggle-competitions>]

Structure of a decision tree

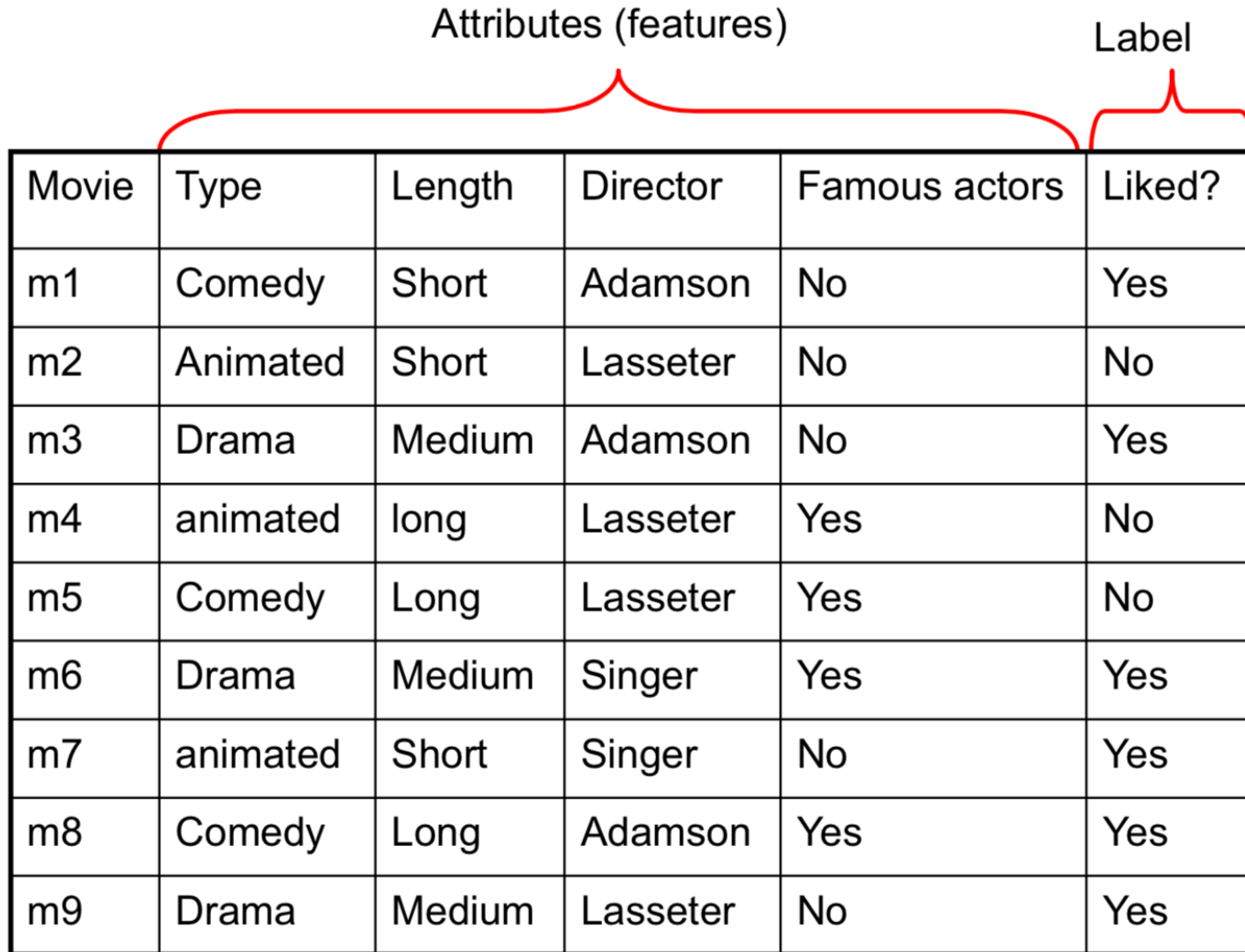
- An advertising example:

- Internal nodes correspond to attributes (features)
- Leafs correspond to classification outcome
- Edges denote assignment



[Slide from Ziv Bar-Joseph]

Dataset



The diagram illustrates a dataset table with two main groups of columns. A red bracket labeled 'Attributes (features)' spans the first five columns: 'Movie', 'Type', 'Length', 'Director', and 'Famous actors'. Another red bracket labeled 'Label' spans the sixth column: 'Liked?'. The table contains 10 rows of data, each representing a movie instance.

Movie	Type	Length	Director	Famous actors	Liked?
m1	Comedy	Short	Adamson	No	Yes
m2	Animated	Short	Lasseter	No	No
m3	Drama	Medium	Adamson	No	Yes
m4	animated	long	Lasseter	Yes	No
m5	Comedy	Long	Lasseter	Yes	No
m6	Drama	Medium	Singer	Yes	Yes
m7	animated	Short	Singer	No	Yes
m8	Comedy	Long	Adamson	Yes	Yes
m9	Drama	Medium	Lasseter	No	Yes

[Slide from Ziv Bar-Joseph]

Building a decision tree

Function BuildTree(n, A) // n : samples (rows), A : attributes

If empty(A) or all $n(L)$ are the same

status = leaf

class = most common class in $n(L)$

$n(L)$: Labels for samples in this set

else

status = internal

$a \leftarrow \text{bestAttribute}(n, A)$

We will discuss this function next

LeftNode = BuildTree($n(a=1)$, $A \setminus \{a\}$)

RightNode = BuildTree($n(a=0)$, $A \setminus \{a\}$)

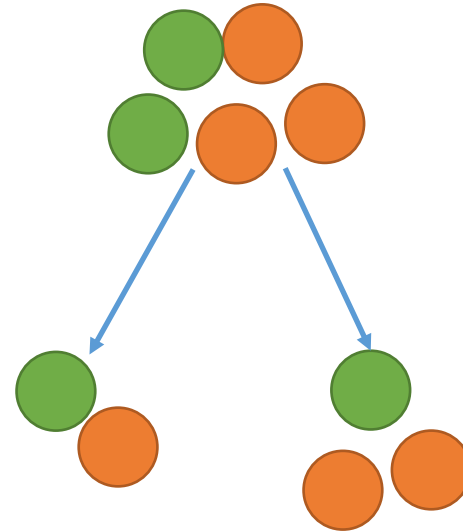
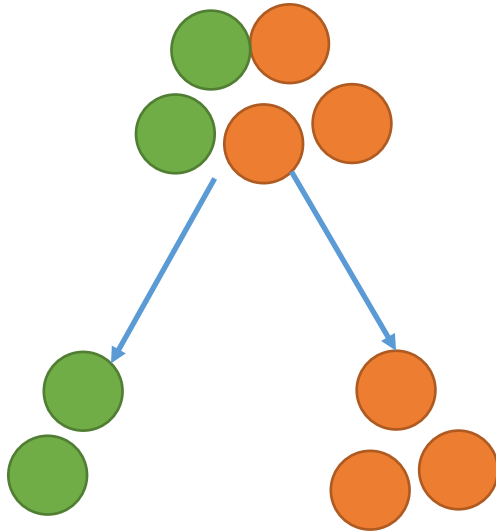
Recursive calls to create left and right subtrees, $n(a=1)$ is the set of samples in n for which the attribute a is 1

end

end

Identifying 'bestAttribute'

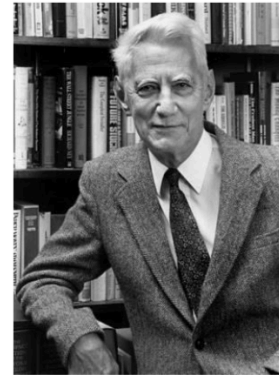
- There are many possible ways to select the best attribute for a given set.
- We will discuss one possible way which is based on information theory and generalizes well to non binary variables.



Entropy

- Quantifies the amount of uncertainty associated with a specific probability distribution
- The higher the entropy, the less confident we are in the outcome
- Definition

$$H(X) = \sum_c -p(X=c) \log_2 p(X=c)$$



Claude Shannon (1916 – 2001), most of the work was done in Bell labs

Entropy

- Definition

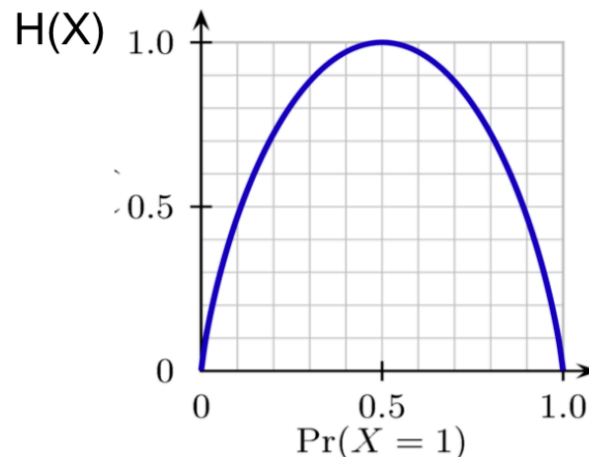
$$H(X) = \sum_i -p(X=i) \log_2 p(X=i)$$

- So, if $P(X=1) = 1$ then

$$\begin{aligned} H(X) &= -p(x=1) \log_2 p(X=1) - p(x=0) \log_2 p(X=0) \\ &= -1 \log 1 - 0 \log 0 = 0 \end{aligned}$$

- If $P(X=1) = .5$ then

$$\begin{aligned} H(X) &= -p(x=1) \log_2 p(X=1) - p(x=0) \log_2 p(X=0) \\ &= -.5 \log_2 .5 - .5 \log_2 .5 = -\log_2 .5 = 1 \end{aligned}$$



[Slide from Ziv Bar-Joseph]

Interpreting entropy

- Entropy can be interpreted from an information standpoint
- Assume both sender and receiver know the distribution. How many bits, on average, would it take to transmit one value?
- If $P(X=1) = 1$ then the answer is 0 (we don't need to transmit anything)
- If $P(X=1) = .5$ then the answer is 1 (either values is equally likely)
- If $0 < P(X=1) < .5$ or $0.5 < P(X=1) < 1$ then the answer is between 0 and 1

Conditional entropy

- Entropy measures the uncertainty in a specific distribution
- What if we know something about the transmission?
- For example, say I want to send the label (liked) when the length is known
- This becomes a conditional entropy problem:
 $H(L_i | L_e=v)$ is the entropy of Liked among movies with length v .

Movie length	Liked?
Short	Yes
Short	No
Medium	Yes
long	No
Long	No
Medium	Yes
Short	Yes
Long	Yes
Medium	Yes

[Slide from Ziv Bar-Joseph]

Conditional entropy: Examples for specific values

- Lets compute $H(Li | Le=v)$
 - 1. $H(Li | Le = S) = .92$
 - 2. $H(Li | Le = M) = 0$
 - 3. $H(Li | Le = L) = .92$

Movie length	Liked?
Short	Yes
Short	No
Medium	Yes
long	No
Long	No
Medium	Yes
Short	Yes
Long	Yes
Medium	Yes

[Slide from Ziv Bar-Joseph]

Conditional entropy

- We can generalize the conditional entropy idea to determine $H(L_i | L_e)$

- Definition: $H(Y | X) = \sum_i P(X = i) H(Y | X = i)$

We explained how to compute this in the previous slides

Movie length	Liked?
Short	Yes
Short	No
Medium	Yes
long	No
Long	No
Medium	Yes
Short	Yes
Long	Yes
Medium	Yes

Conditional entropy: Example

$$H(Y | X) = \sum_i P(X = i) H(Y | X = i)$$

○ Lets compute $H(L_i | L_e)$

$$\begin{aligned} H(L_i | L_e) &= P(L_e = S) H(L_i | L_e=S) + \\ &\quad P(L_e = M) H(L_i | L_e=M) + \\ &\quad P(L_e = L) H(L_i | L_e=L) = \\ &= 1/3 \cdot .92 + 1/3 \cdot 0 + 1/3 \cdot .92 = \\ &= 0.61 \end{aligned}$$

we already computed:

$$H(L_i | L_e = S) = .92$$

$$H(L_i | L_e = M) = 0$$

$$H(L_i | L_e = L) = .92$$

Movie length	Liked?
Short	Yes
Short	No
Medium	Yes
long	No
Long	No
Medium	Yes
Short	Yes
Long	Yes
Medium	Yes

[Slide from Ziv Bar-Joseph]

Information gain

- How much do we gain (in terms of reduction in entropy) from knowing one of the attributes
- In other words, what is the reduction in entropy from this knowledge
- Definition: $IG(Y|X) = H(Y) - H(Y|X)$
 - $IG(X|Y)$ is always ≥ 0 Proof: Jensen inequality

Where we are

- We were looking for a good criteria for selecting the best attribute for a node split
- We defined the entropy, conditional entropy and information gain
- We will now use information gain as our criteria for a good split
- That is, `bestAttribute` will return the attribute that maximizes the information gain at each node

Building a decision tree

Function BuildTree(n,A) // n: samples (rows), A: attributes

 If empty(A) or all n(L) are the same

 status = leaf

 class = most common class in n(L)

 else

 status = internal

$a \leftarrow \text{bestAttribute}(n,A)$

 LeftNode = BuildTree(n(a=1), $A \setminus \{a\}$)

 RightNode = BuildTree(n(a=0), $A \setminus \{a\}$)

 end

end

Based on information gain



Example: Root attribute

- $P(Li=yes) = 2/3$

- $H(Li) = .91$

- $H(Li | T) = 0.61$

- $H(Li | Le) = 0.61$

- $H(Li | D) = 0.36$

- $H(Li | F) = 0.85$

- $IG(Li | T) = .91 - .61 = 0.3$

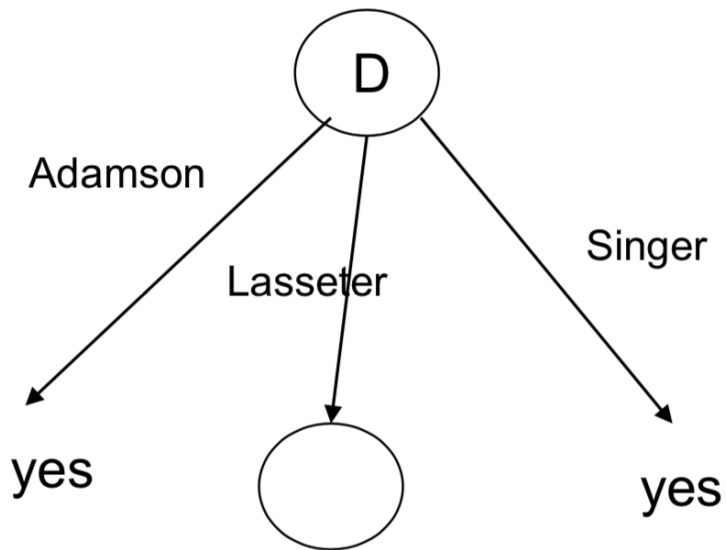
- $IG(Li | Le) = .91 - .61 = 0.3$

- $IG(Li | D) = .91 - .36 = 0.55$

- $IG(Li | F) = .91 - .85 = 0.06$

Movie	Type	Length	Director	Famous actors	Liked ?
m1	Comedy	Short	Adamson	No	Yes
m2	Animated	Short	Lasseter	No	No
m3	Drama	Medium	Adamson	No	Yes
m4	animated	long	Lasseter	Yes	No
m5	Comedy	Long	Lasseter	Yes	No
m6	Drama	Medium	Singer	Yes	Yes
M7	animated	Short	Singer	No	Yes
m8	Comedy	Long	Adamson	Yes	Yes
m9	Drama	Medium	Lasseter	No	Yes

Building a tree

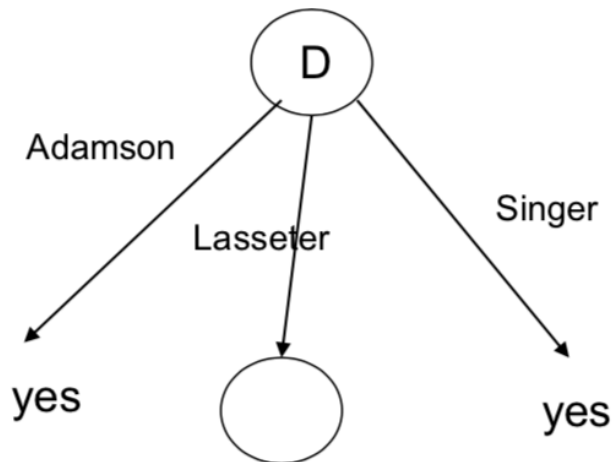


Movie	Type	Length	Director	Famous actors	Liked ?
m1	Comedy	Short	Adamson	No	Yes
m2	Animated	Short	Lasseter	No	No
m3	Drama	Medium	Adamson	No	Yes
m4	animated	long	Lasseter	Yes	No
m5	Comedy	Long	Lasseter	Yes	No
m6	Drama	Medium	Singer	Yes	Yes
M7	animated	Short	Singer	No	Yes
m8	Comedy	Long	Adamson	Yes	Yes
m9	Drama	Medium	Lasseter	No	Yes

[Slide from Ziv Bar-Joseph]

Building a tree

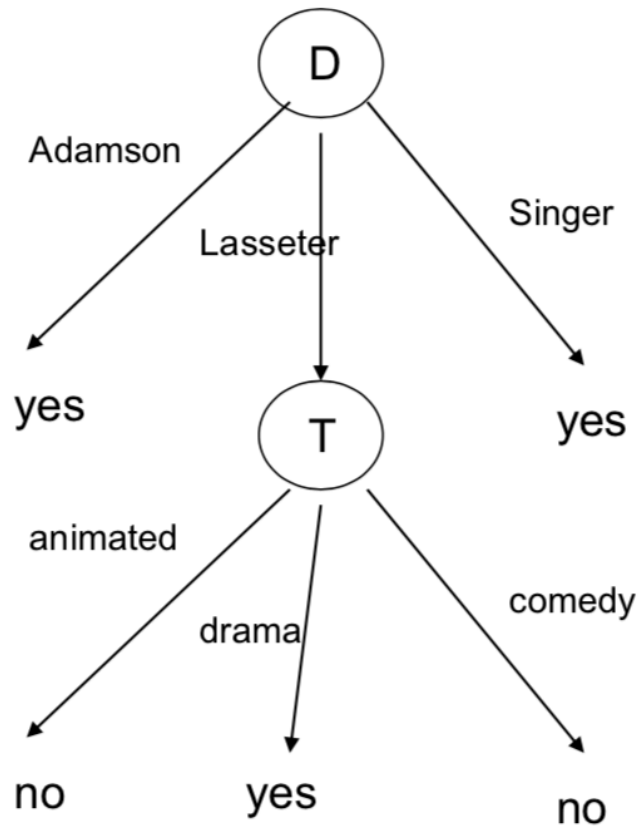
- We only need to focus on the records (samples) associated with this node
- We eliminated the 'director' attribute. All samples have the same director
- $P(Li=yes) = \frac{1}{4}$, $H(Li) = .81$
- $H(Li | T) = 0$, $H(Li | Le) = 0$, $H(Li | F) = 0.5$
- $IG(Li | T) = 0.81$, $IG(Li | Le) = 0.81$, $IG(Li | F) = .31$



Movie	Type	Length	Director	Famous actors	Liked ?
m2	Animated	Short	Lasseter	No	No
m4	animated	Long	Lasseter	Yes	No
m5	Comedy	Long	Lasseter	Yes	No
m9	Drama	Medium	Lasseter	No	Yes

[Slide from Ziv Bar-Joseph]

Final tree



Movie	Type	Length	Famous actors	Liked ?
m2	Animated	Short	No	No
m4	animated	long	Yes	No
m5	Comedy	Long	Yes	No
m9	Drama	Medium	No	Yes

[Slide from Ziv Bar-Joseph]

Continuous values

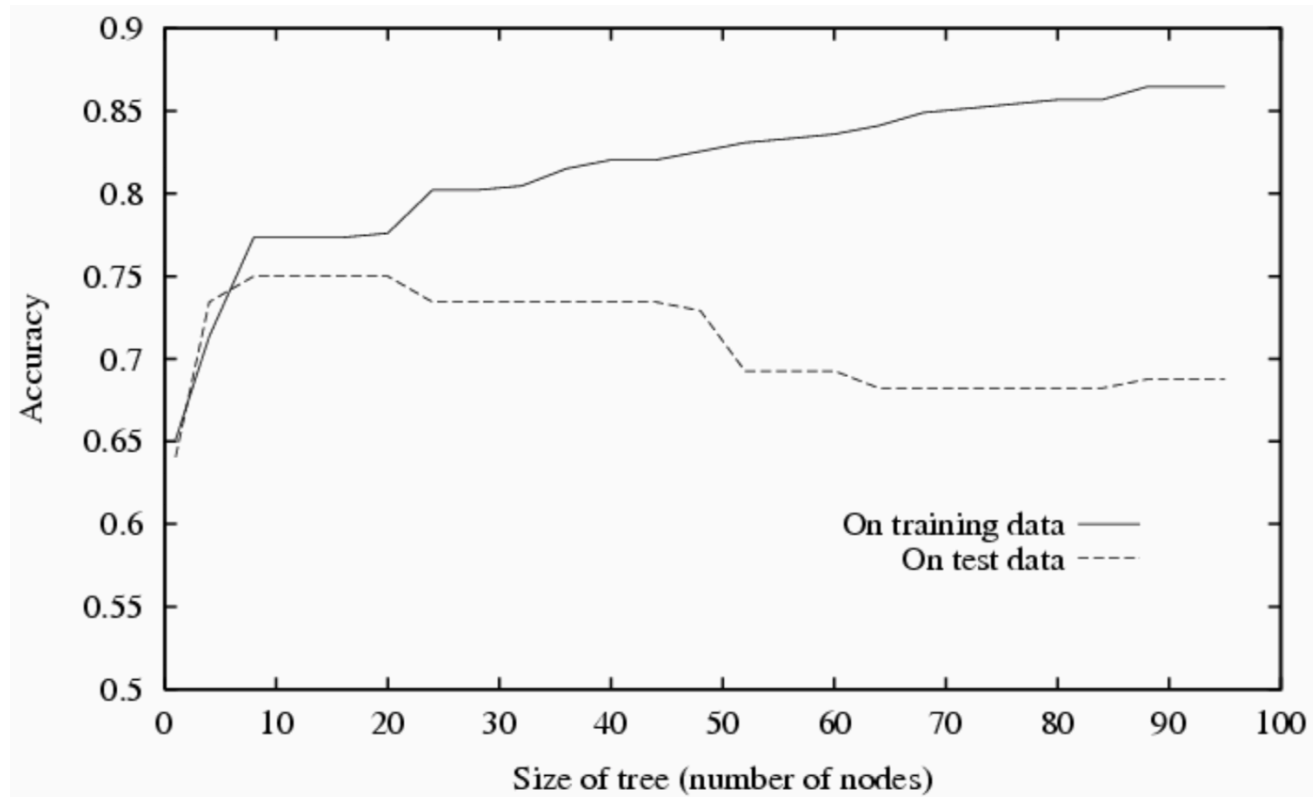
- Either use threshold to turn into binary or discretize
- It's possible to compute information gain for all possible thresholds (there are a finite number of training samples)
- Harder if we wish to assign more than two values (can be done recursively)

Additional points

- The algorithm we gave reaches homogenous nodes (or runs out of attributes)
- This is dangerous: For datasets with many (non relevant) attributes the algorithm will continue to split nodes
- This will lead to overfitting!

Overfitting

- Training/empirical error: $\sum_{i=1}^n I(f(X_i) \neq Y_i) / n$
- Error over whole data distribution: $R(f) = P(f(X) \neq Y)$



[Slide from Tom Mitchell]

Avoiding overfitting: Tree pruning

- One possible way in decision tree to avoid overfitting
 - Other ways: Do not grow tree beyond some maximum depth; Do not split if splitting criterion (e.g. mutual information) is below some threshold; Stop growing when the split is not statistically significant
- Split data into train and testing set
- Build tree using training set
- For all internal nodes (starting at the root)
 - remove subtree rooted at node
 - assign class to be the most common among training set
 - check test data error
 - if error is lower, keep change
 - otherwise restore subtree, repeat for all nodes in subtree

Random forest: bootstrap

- A collection of decision trees: ensemble methods
- For each tree we select a subset of the attributes (recommended square root of $|A|$), a subset of samples, and build tree using just these attributes
- An input sample is classified using majority voting

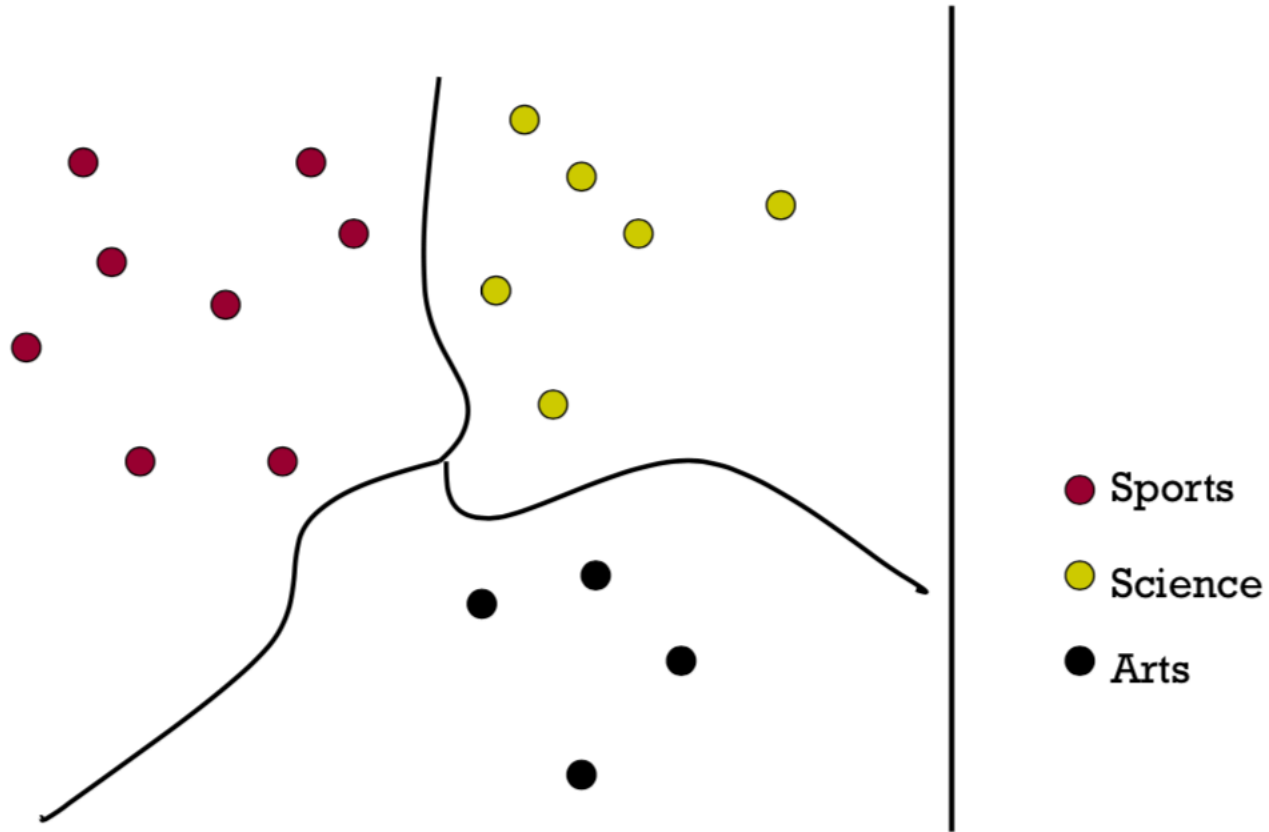
Revisit Document Classification: Vector Space Representation

- Each document is a vector, one component for each term (= word).

	Doc 1	Doc 2	Doc 3	...
Word 1	3	0	0	...
Word 2	0	8	1	...
Word 3	12	1	10	...
...	0	1	3	...
...	0	0	0	...

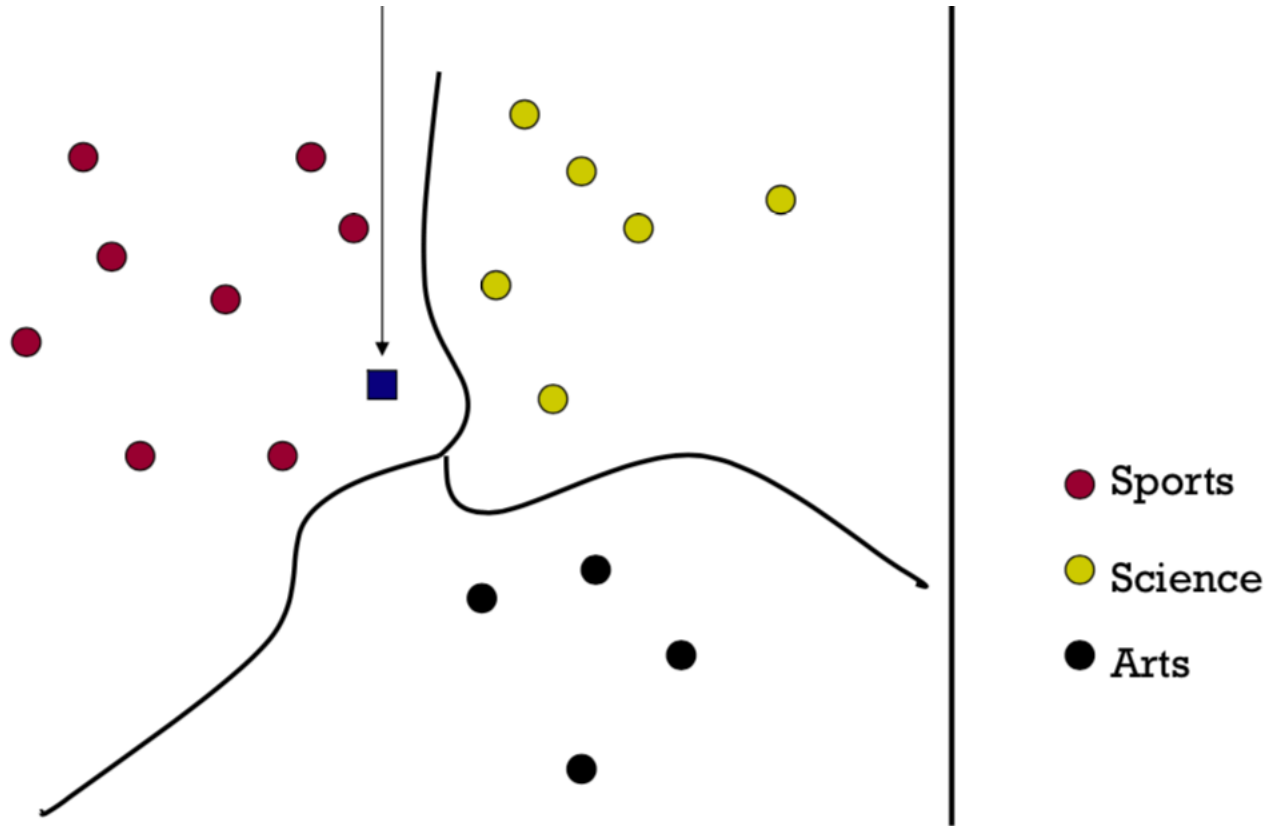
- Normalize to unit length.
- High-dimensional vector space:
 - Terms are axes, 10,000+ dimensions, or even 100,000+
 - Docs are vectors in this space

Classes in a Vector Space



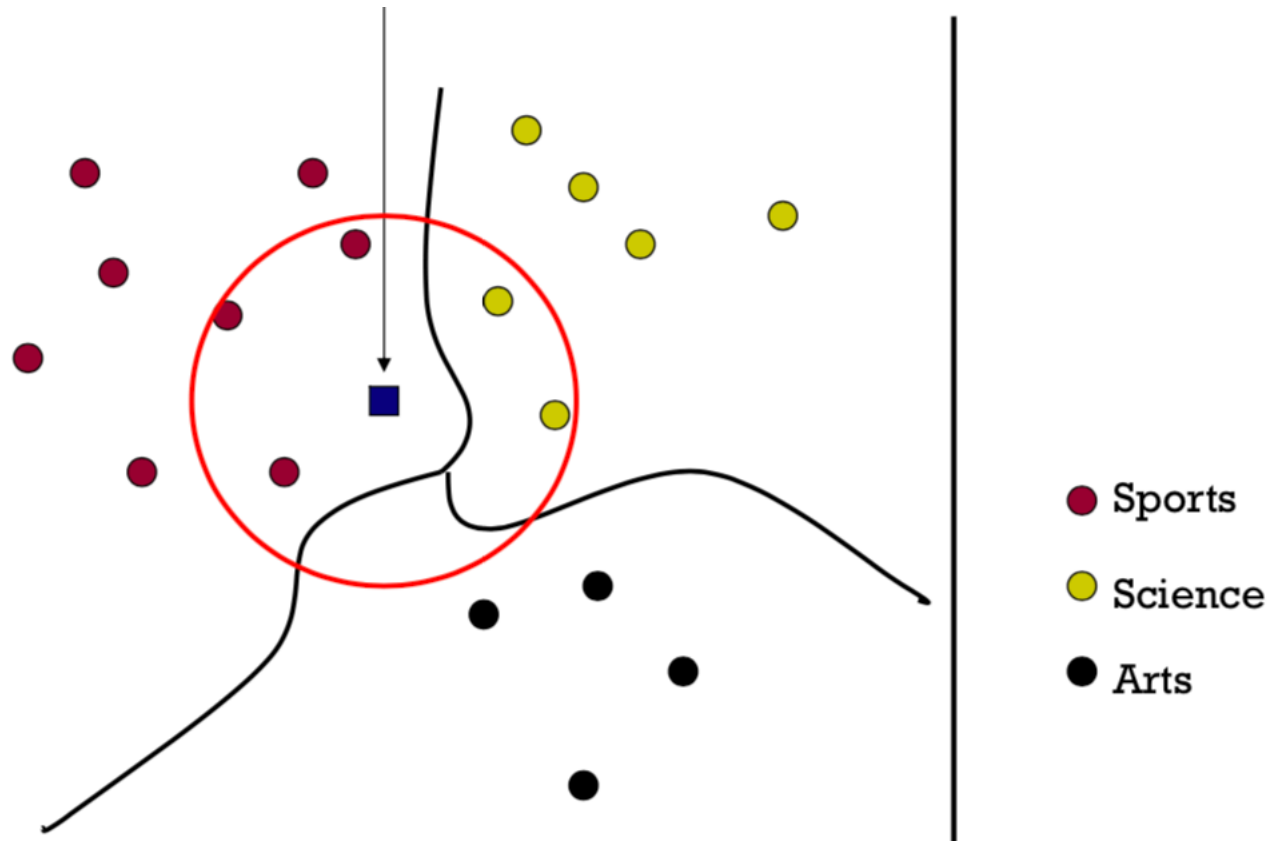
[Slide from Eric Xing]

Test Document = ?



[Slide from Eric Xing]

K-Nearest Neighbor (kNN) classifier



[Slide from Eric Xing]

kNN is an instance of Instance-Based Learning

- What makes an Instance-Based Learner?
 - A distance metric
 - How many nearby neighbors to look at?
 - A weighting function (optional)

Euclidean Distance Metric

$$D(x, x') = \sqrt{\sum_i \sigma_i^2 (x_i - x'_i)^2}$$

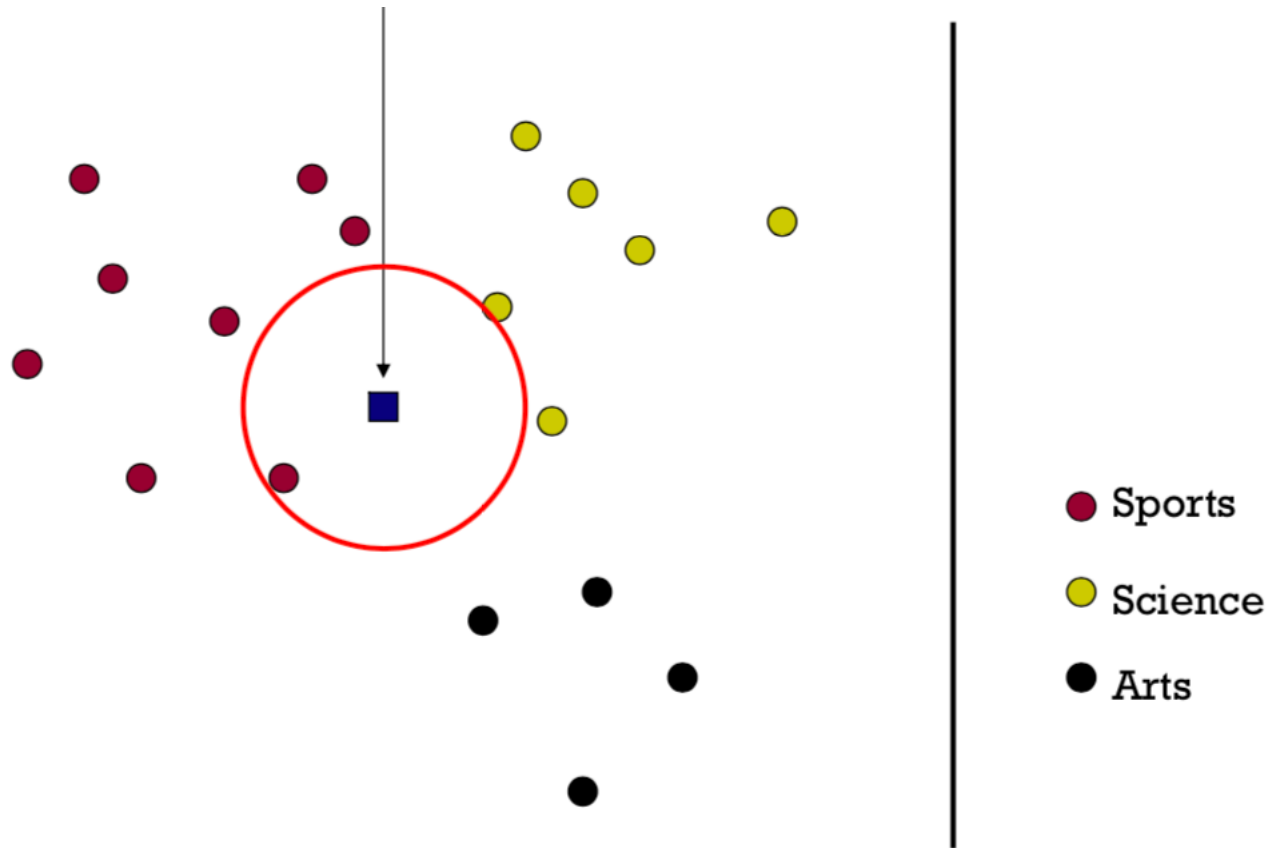
Or equivalently,

$$D(x, x') = \sqrt{(x - x')^T \Sigma (x - x')}$$

Other metrics:

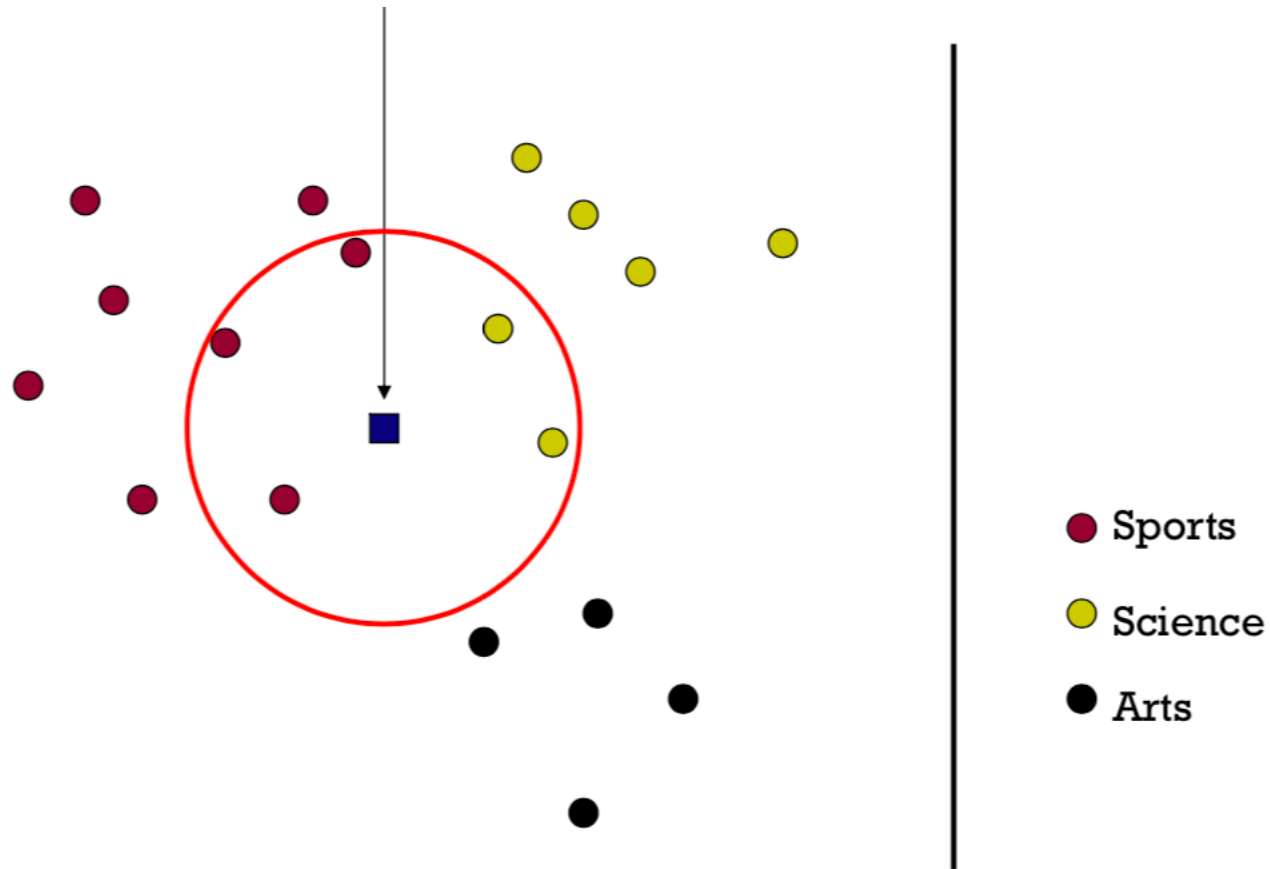
- L_1 norm: $|x - x'|$
- L_∞ norm: $\max |x - x'|$ (elementwise ...)
- Mahalanobis: where Σ is full, and symmetric
- Correlation
- Angle
- Hamming distance, Manhattan distance
- ...

1-Nearest Neighbor (kNN) classifier



[Slide from Eric Xing]

5-Nearest Neighbor (kNN) classifier



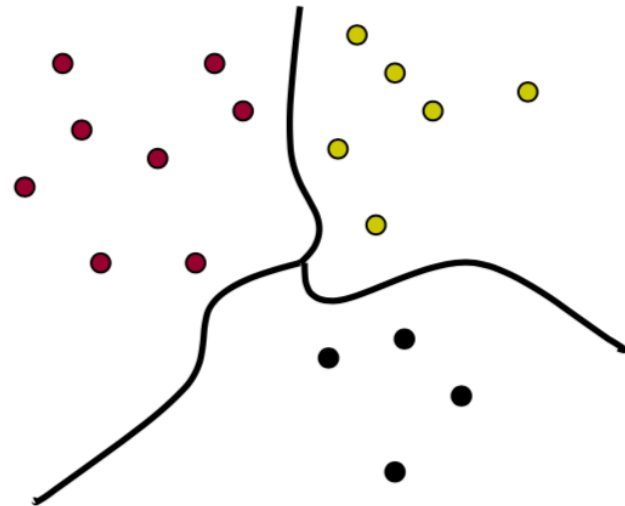
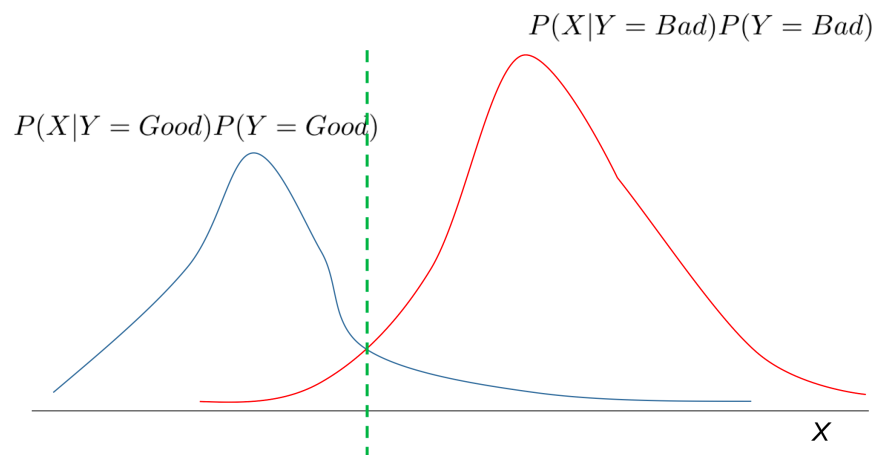
[Slide from Eric Xing]

Nearest-Neighbor Learning Algorithm

- Learning is just storing the representations of the training examples in D .
- Testing instance x :
 - Compute similarity between x and all examples in D .
 - Assign x the category of the most similar example in D .
- Does not explicitly compute a generalization or category prototypes.
- Also called:
 - Case-based learning
 - Memory-based learning
 - Lazy learning

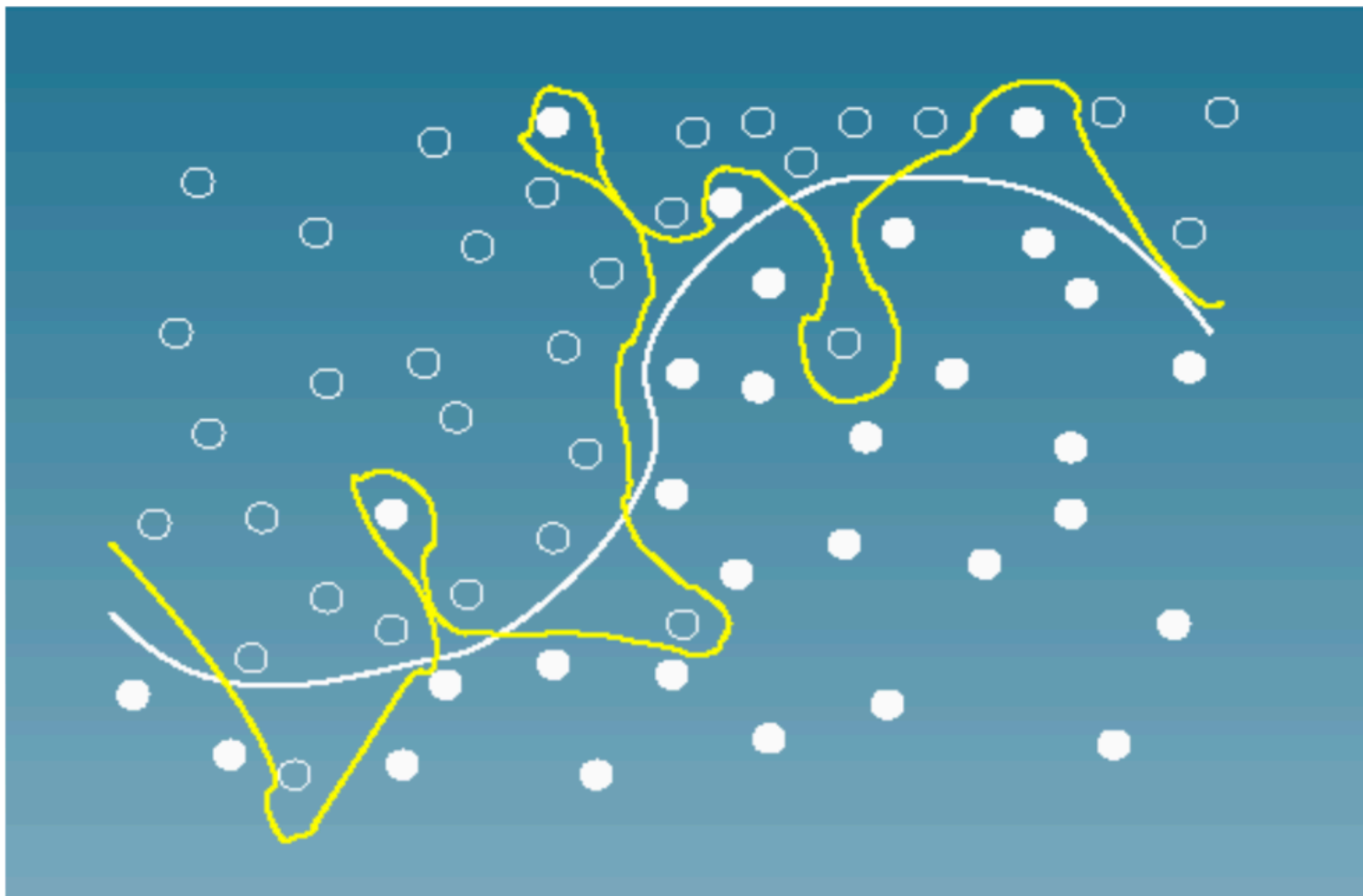
kNN Is Close to Optimal

- Cover and Hart 1967
- Asymptotically, the error rate of 1-nearest-neighbor classification is less than twice the **Bayes rate** [error rate of classifier knowing model that generated data]
- In particular, asymptotic error rate is 0 if Bayes rate is 0.
- Decision boundary:



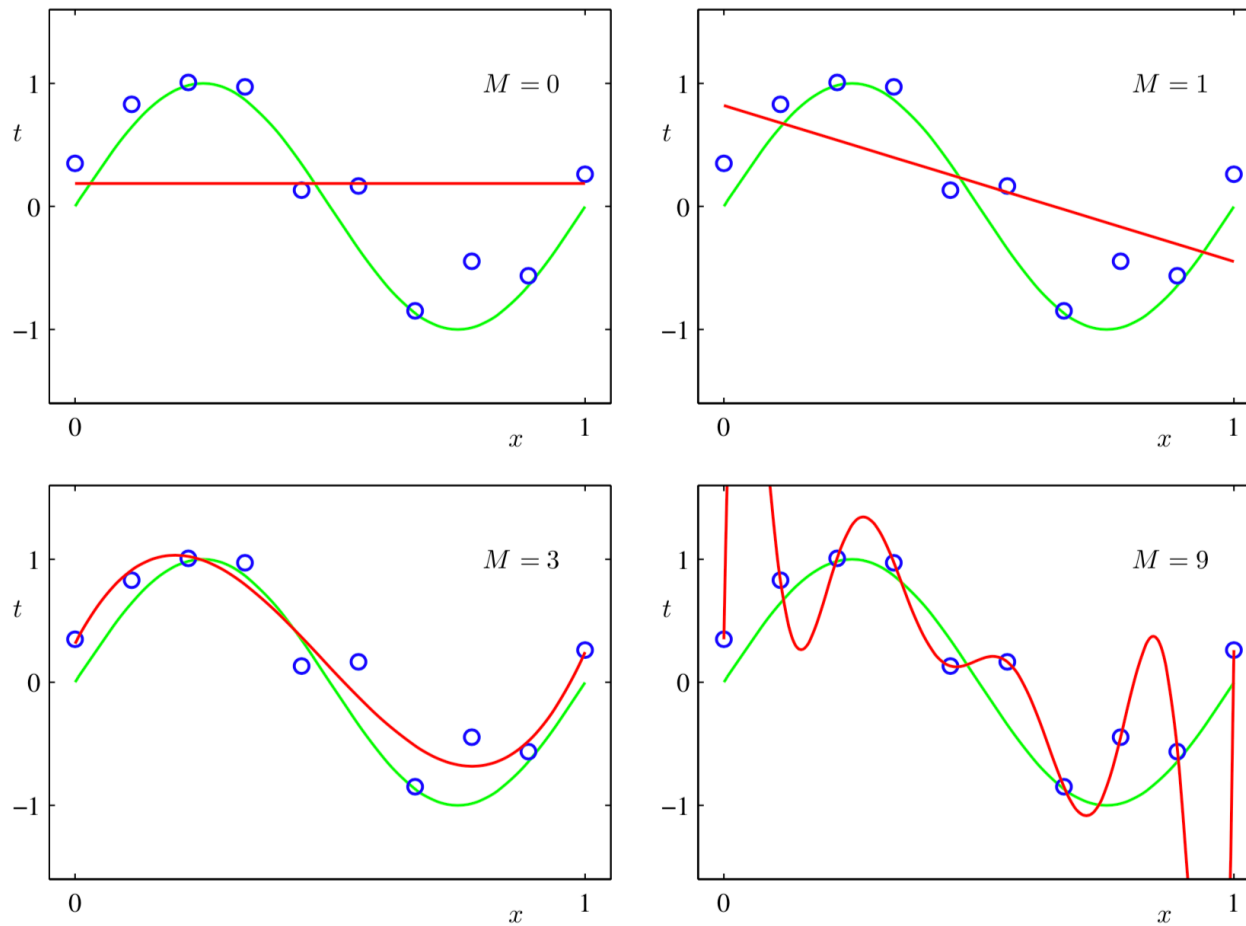
[Slide from Eric Xing]

Overfitting



[Slide from Eric Xing]

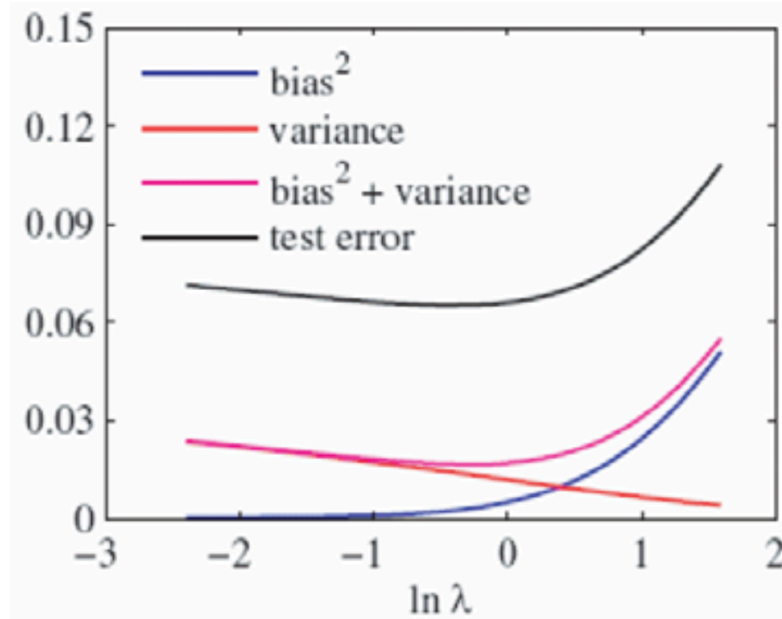
Another example



[Figure from Christopher Bishop]

Bias-variance decomposition

- Now let's look more closely into two sources of errors in an functional approximator:



- In the following we show the Bias-variance decomposition using LR as an example.

Loss functions for regression

Let t be the true (target) output and $y(x)$ be our estimate. The expected squared loss is

$$\begin{aligned} E(L) &= \iint L(t, y(x)) p(x, t) dx dt \\ &= \iint (t - y(x))^2 p(x, t) dx dt \end{aligned}$$

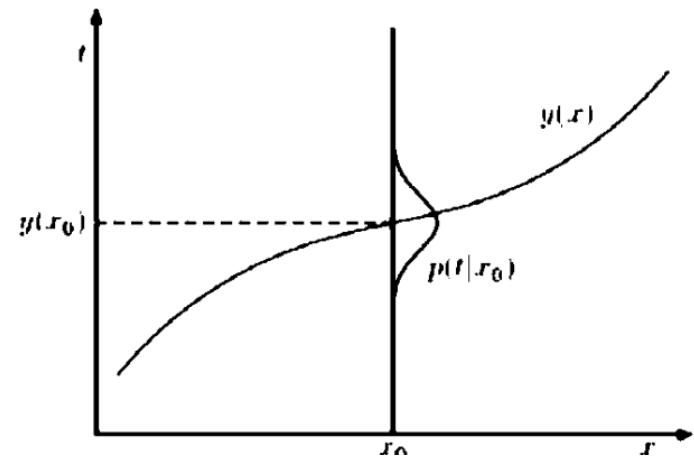
Our goal is to choose $y(x)$ that minimize $E(L)$:

- Calculus of variations:

$$\frac{\partial E(L)}{\partial y(x)} = 2 \int (t - y(x)) p(x, t) dt = 0$$

$$\int y(x) p(x, t) dt = \int t p(x, t) dt$$

$$y^*(x) = \int \frac{t p(x, t)}{p(x)} dt = \int t p(t | x) dt = E_{t|x}[t] = E[t | x]$$



Expected loss

Let $h(x) = E[t|x]$ be the **optimal** predictor, and $y(x)$ our actual predictor, which will incur the following expected loss

$$\begin{aligned} E(y(x) - t)^2 &= \int (y(x) - h(x) + h(x) - t)^2 p(x, t) dx dt \\ &= \int (y(x) - h(x))^2 + 2(y(x) - h(x))(h(x) - t) + (h(x) - t)^2 p(x, t) dx dt \\ &= \int (y(x) - h(x))^2 p(x) dx + \int (h(x) - t)^2 p(x, t) dx dt \end{aligned}$$

- $\int (h(x) - t)^2 p(x, t) dx dt$ is a noisy term, and we can do no better than this. Thus it is a lower bound of the expected loss.
- The other part of the error come from $\int (y(x) - h(x))^2 p(x) dx$, and let's take a close look of it.
- We will assume $y(x) = y(x|w)$ is a parametric model and the parameters w are fit to a training set D . (thus we write $y(x; D)$)

Bias-variance decomposition

For one data set D and one test point x

- since the predictor y depend on the data training data D , write $E_D[y(x,D)]$ for the expected predictor over the ensemble of datasets, then (using the same trick) we have:

$$\begin{aligned}(y(x; D) - h(x))^2 &= (y(x; D) - E_D[y(x; D)] + E_D[y(x; D)] - h(x))^2 \\&= (y(x; D) - E_D[y(x; D)])^2 + (E_D[y(x; D)] - h(x))^2 \\&\quad + 2(y(x; D) - E_D[y(x; D)])(E_D[y(x; D)] - h(x))\end{aligned}$$

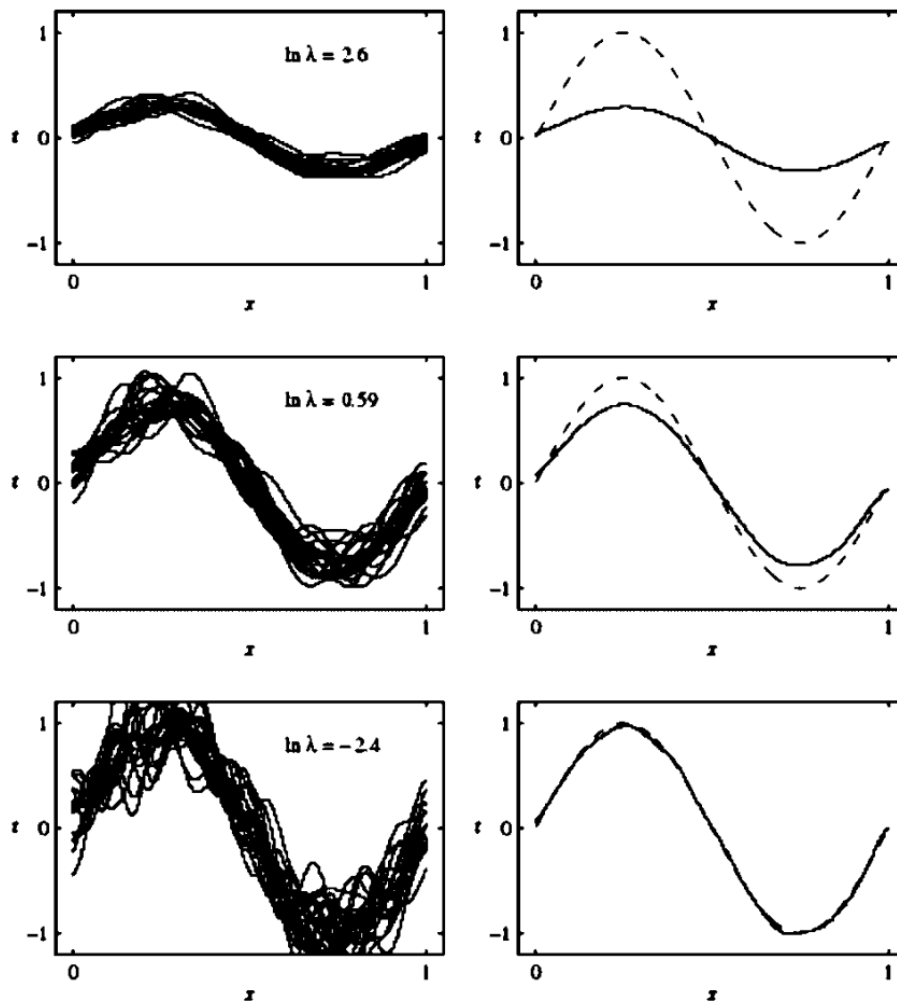
- Surely this error term depends on the training data, so we take an expectation over them:

$$E_D[(y(x; D) - h(x))^2] = (E_D[y(x; D)] - h(x))^2 + E_D[(y(x; D) - E_D[y(x; D)])^2]$$

Putting things together:

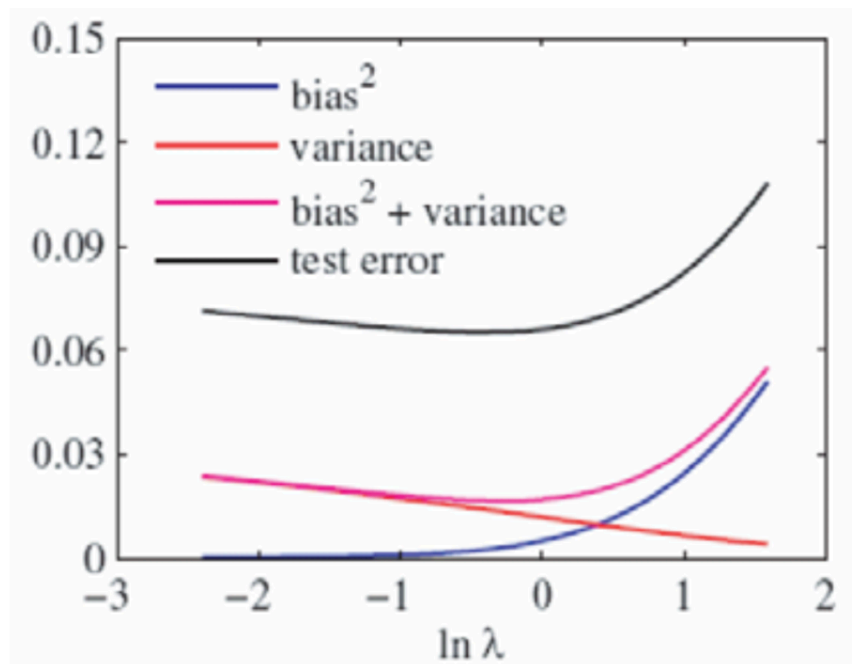
$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

Bias-variance tradeoff: Regularized Regression



- λ is a "regularization" terms in LR, the smaller the λ , is more complex the model (why?)
 - Simple (highly regularized) models have low variance but high bias.
 - Complex models have low bias but high variance.
- You are inspecting an empirical average over 100 training set.
- The actual E_D can not be computed

Bias²+variance vs regularizer



- Bias²+variance predicts (shape of) test error quite well.
- However, bias and variance cannot be computed since it relies on knowing the true distribution of x and t (and hence $h(x) = E[t|x]$).

Model Selection

- Suppose we are trying select among several different models for a learning problem.
- Examples:
 - polynomial regression
 - Model selection: we wish to automatically and objectively decide if M should be, say, 0, 1, ..., or 10.
 - Etc...
- The Problem:

Given model family $\mathcal{F} = \{M_1, M_2, \dots, M_I\}$, find $M_i \in \mathcal{F}$ s.t.

$$M_i = \arg \max_{M \in \mathcal{F}} J(D, M)$$

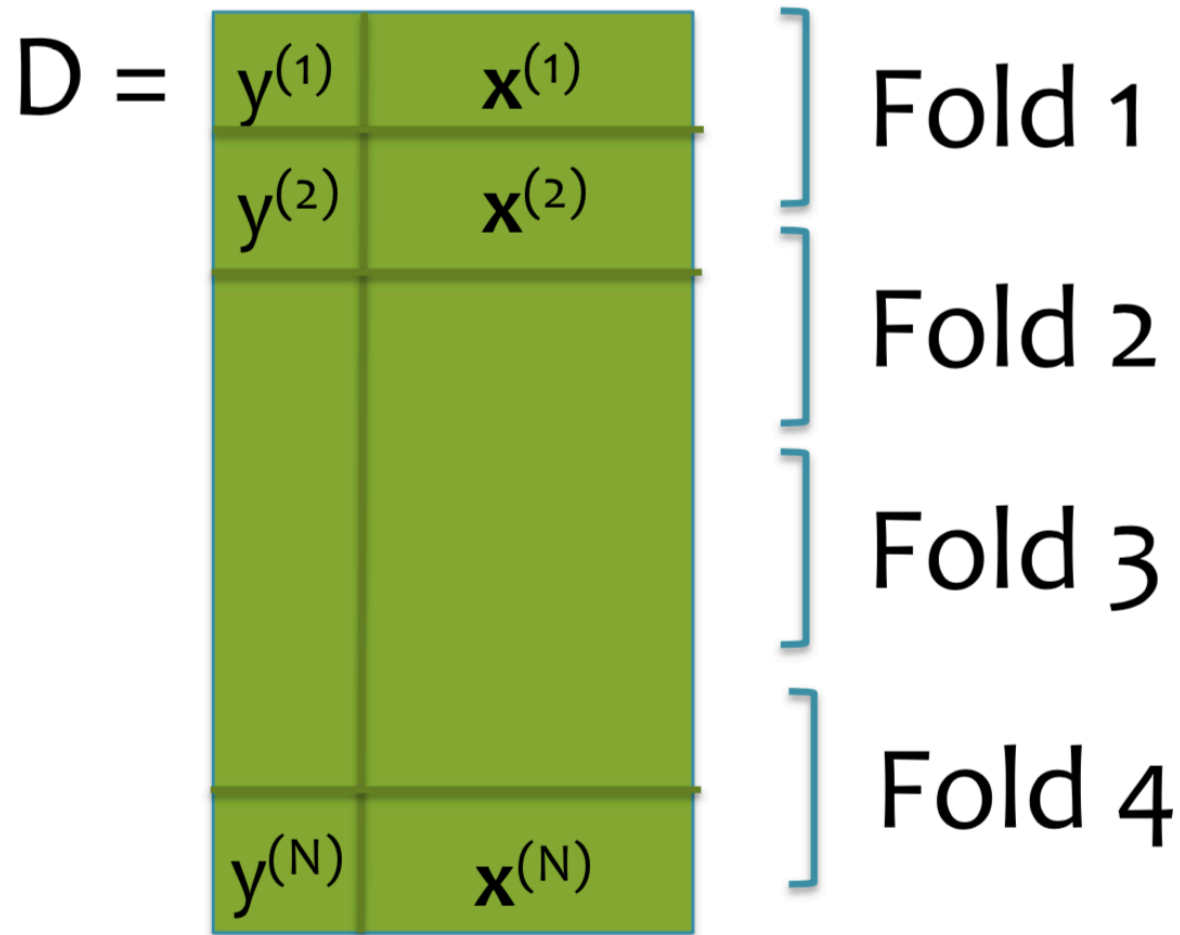
Cross Validation

We are given training data D and test data D_{test} , and we would like to fit this data with a model $p_i(x; \theta)$ from the family \mathcal{F} (e.g, an LR), which is indexed by i and parameterized by θ .

K -fold cross-validation (CV)

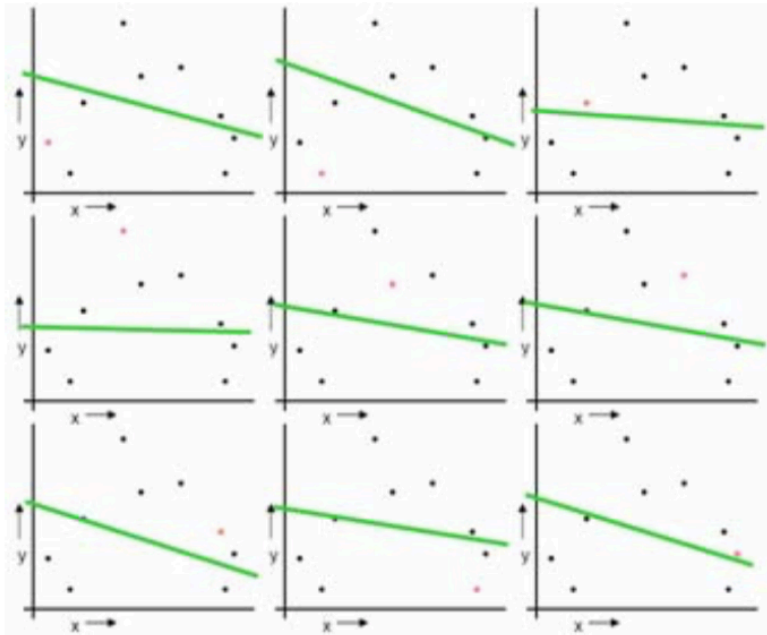
- Set aside αN samples of D (where $N = |D|$). This is known as the held-out data and will be used to evaluate different values of i .
- For each candidate model i , fit the optimal hypothesis $p_i(x; \theta^*)$ to the remaining $(1-\alpha)N$ samples in D (i.e., hold i fixed and find the best θ).
- Evaluate each model $p_i(x; \theta^*)$ on the held-out data using some pre-specified risk function.
- Repeat the above **K times**, choosing a **different** held-out data set each time, and the scores are averaged for each model $p_i(\cdot)$ over all held-out data set. This gives an estimate of the risk curve of models over different i .
- For the model with the lowest risk, say $p_{i^*}(\cdot)$, we use all of D to find the parameter values for $p_{i^*}(x; \theta^*)$.

Cross Validation

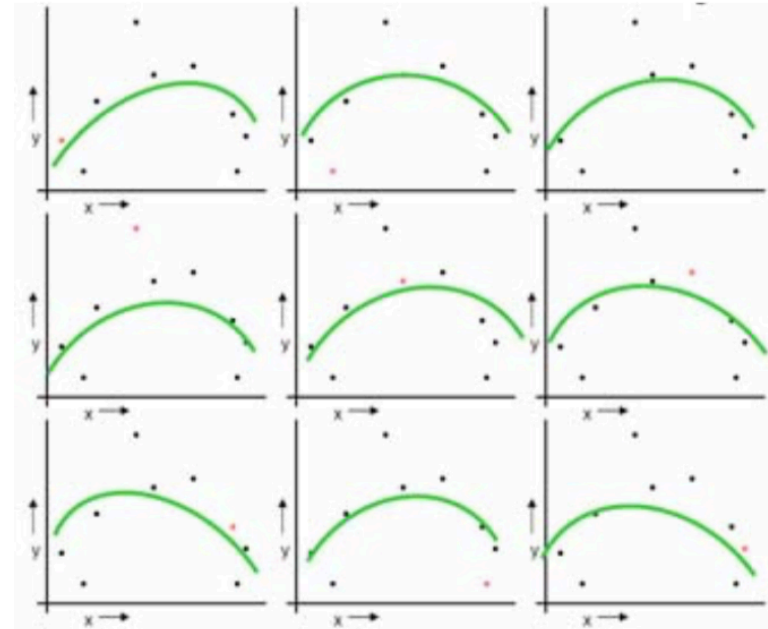


Example

- When $\alpha=1/N$, the algorithm is known as Leave-One-Out Cross-Validation (LOOCV)



$$MSE_{LOOCV}(M_1)=2.12$$



$$MSE_{LOOCV}(M_2)=0.962$$

Practical issues for CV

- How to decide the values for K and α
 - Commonly used $K=10$ and $\alpha=0.1$.
 - When data sets are small relative to the number of models that are being evaluated, we need to decrease α and increase K
- Bias-variance trade-off
 - Small α usually lead to low bias. In principle, $LOOCV$ provides an almost unbiased estimate of the generalization ability of a classifier, especially when the number of the available training samples is severely limited.
 - Large α can reduce variance, but will lead to under-use of data, and causing high-bias.
- One important point is that the test data D_{test} is never used in CV, because doing so would result in overly (indeed dishonest) optimistic accuracy rates during the testing phase.

Regularization

- More in lecture 1...

The posterior distribution of θ

$$p(\theta|D) \propto \exp\left\{-\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \theta^T x_i)^2\right\} \times \exp\left\{-\theta^T \theta / 2\tau^2\right\}$$

This leads to a new objective

$$\begin{aligned} l_{MAP}(\theta; D) &= -\frac{1}{2\sigma^2} \frac{1}{2} \sum_{i=1}^n (y_i - \theta^T \mathbf{x}_i)^2 - \frac{1}{\tau^2} \frac{1}{2} \sum_{k=1}^K \theta_k^2 \\ &= l(\theta; D) - \lambda \|\theta\| \end{aligned}$$

- This is L_2 regularized LR! --- a MAP estimation of θ
- What about L_1 regularized LR! (homework)

How to choose λ .

- cross-validation!

AIC and BIC

- Akaike information criterion (AIC):

$$\text{AIC} = 2k - 2 \ln(\hat{L})$$

- Bayesian information criterion (BIC):

$$\text{BIC} = \ln(n)k - 2 \ln(\hat{L})$$

Take home message

- Decision tree is a discriminative classifier
 - Built recursively based on maximizing information gain
 - Prevent overfitting through pruning and bootstrap
- kNN is close to optimal (asymptotically)
- Tradeoff between bias and variance of models
$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$
- Methods to reduce overfitting
 - Cross-validation
 - Regularization
 - AIC, BIC
 - Ensemble methods: bootstrap

References

- Eric Xing, Tom Mitchell. 10701 Introduction to Machine Learning:
<http://www.cs.cmu.edu/~epxing/Class/10701-06f/>
- Eric Xing, Ziv Bar-Joseph. 10701 Introduction to Machine Learning:
<http://www.cs.cmu.edu/~epxing/Class/10701/>
- Matt Gormley. 10601 Introduction to Machine Learning:
<http://www.cs.cmu.edu/~mgormley/courses/10601/index.html>
- Christopher M. Bishop. Pattern recognition and Machine Learning
- Wikipedia