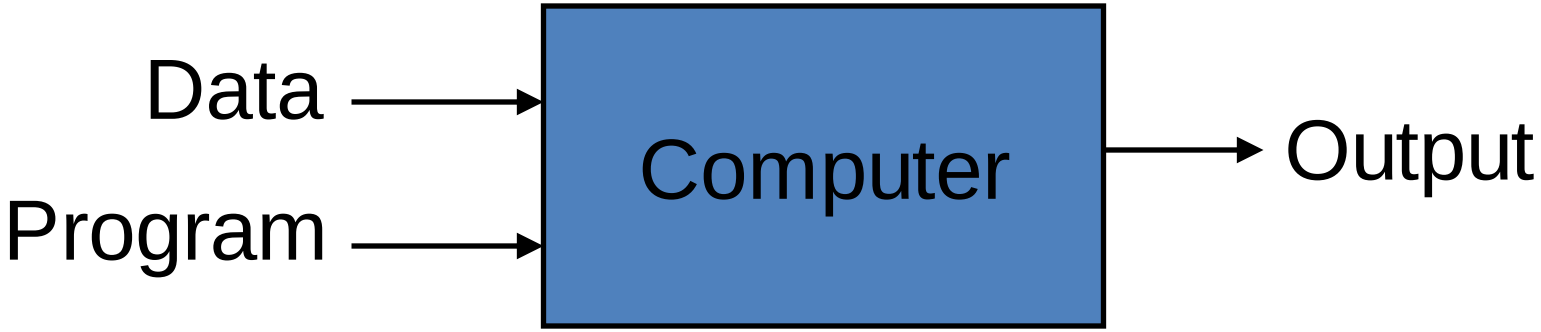
Machine Learning
in 90 minutes

based on slides by Mausam, James Hays, which are based on …

Traditional Programming

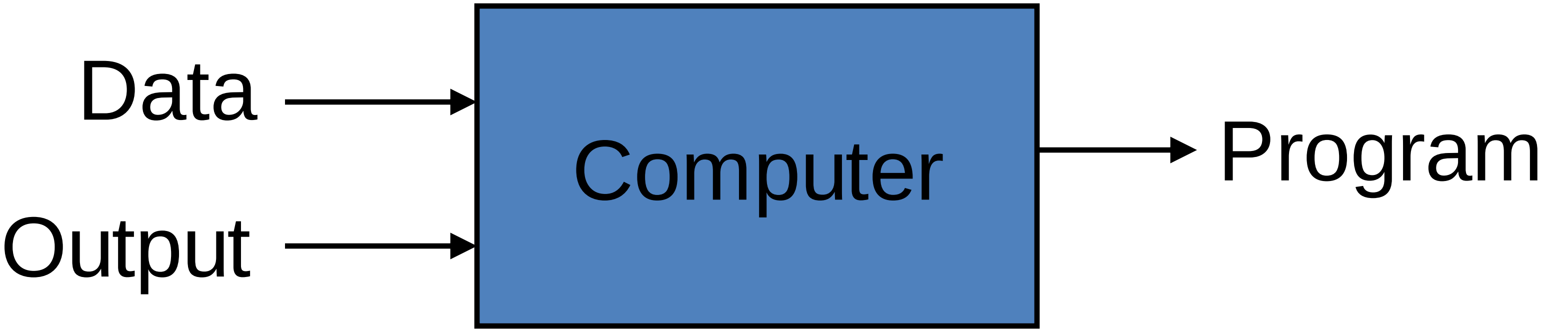

Data

Program

Computer

Output

Machine Learning


Data

Output

Computer

Program

Inductive Learning

- **Given** examples of a function *(X, F(X))*
- **Predict** function *F(X)* for new examples *X*
  - Discrete *F(X)*: Classification
  - Continuous *F(X)*: Regression
  - *F(X)* = Probability(*X*): Probability estimation

Training Data Versus Test

- Terms: ‘data’, ‘examples’, and ‘instances’ used interchangeably

- Training data: data where the labels are given
- Test data: data where the labels are known but not given

Which do you use to measure performance?

Cross validation…

Basic Setup

- **Input:**
  - Labeled training examples
  - Hypothesis space H

- **Output:** hypothesis h in H that is consistent with the training data & (hopefully) correctly classifies test data.

The ‘new’ Machine Learning

Old	New
Small data sets (100s of examples)	Massive (10^6 to 10^10)
Hand-labeled data	Automatically labeled; semi supervised; labeled by “crowds”
Hand-coded algorithms	WEKA package downloaded over 1,000,000 times

ML in a Nutshell

- 10^5 machine learning algorithms
- Hundreds new every year
- Every algorithm has three components:
  - **Hypothesis space—possible outputs**
  - **Search strategy---strategy for exploring space**
  - **Evaluation**

Hypothesis Space (Representation)

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

Metrics for Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Etc.

Based on Data

Search Strategy

- Greedy (depth-first, best-first, hill climbing)
- Exhaustive
- Optimize an objective function
- More…

Types of Learning

- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

Supervised Learning

- **Given:** Training examples $\langle \mathbf{x}, f(\mathbf{x}) \rangle$ for some unknown function f.
- **Find:** A good approximation to f.

Example Applications

- **Credit risk assessment**

  $\mathbf{x}$: Properties of customer and proposed purchase.

  $f(\mathbf{x})$: Approve purchase or not.

- **Disease diagnosis**

  $\mathbf{x}$: Properties of patient (symptoms, lab tests)

  $f(\mathbf{x})$: Disease (or maybe, recommended therapy)

- **Face recognition**

  $\mathbf{x}$: Bitmap picture of person's face

  $f(\mathbf{x})$: Name of the person.

- **Automatic Steering**

  $\mathbf{x}$: Bitmap picture of road surface in front of car.

  $f(\mathbf{x})$: Degrees to turn the steering wheel.

Appropriate Applications for Supervised Learning

- **Situations where there is no human expert**

  $\mathbf{x}$: Bond graph for a new molecule.

  $f(\mathbf{x})$: Predicted binding strength to AIDS protease molecule.

- **Situations where humans can perform the task but can't describe how they do it.**

  $\mathbf{x}$: Bitmap picture of hand-written character

  $f(\mathbf{x})$: Ascii code of the character

- **Situations where the desired function is changing frequently**

  $\mathbf{x}$: Description of stock prices and trades for last 10 days.

  $f(\mathbf{x})$: Recommended stock transactions

- **Situations where each user needs a customized function f**

  $\mathbf{x}$: Incoming email message.

  $f(\mathbf{x})$: Importance score for presenting to user (or deleting without presenting).

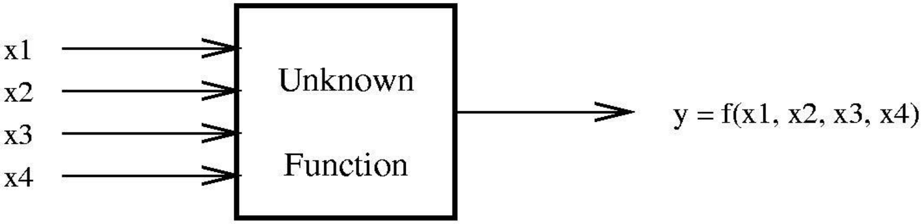
Why Learning?

- Learning is essential for unknown environments
  - e.g., when designer lacks omniscience
- Learning is necessary in dynamic environments
  - Agent can adapt to changes in environment not foreseen at design time
- Learning is useful as a system construction method
  - Expose the agent to reality rather than trying to approximate it through equations etc.
- Learning modifies the agent's decision mechanisms to improve performance

Terminology

- **Training example.** An example of the form $\langle \mathbf{x}, f(\mathbf{x}) \rangle$.
- **Target function (target concept).** The true function f.
- **Hypothesis**. A proposed function h believed to be similar to f.
- **Concept**. A boolean function. Examples for which $f(\mathbf{x}) = 1$ are called **positive examples** or **positive instances** of the concept. Examples for which $f(\mathbf{x}) = 0$ are called **negative examples** or **negative instances.**
- **Classifier**. A discrete-valued function. The possible values $f(\mathbf{x}) \in \{1, \ldots, K\}$ are called the **classes** or **class labels**.
- **Hypothesis Space**. The space of all hypotheses that can, in principle, be output by a learning algorithm.
- **Version Space**. The space of all hypotheses in the hypothesis space that have not yet been ruled out by a training example.

A Learning Problem



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Hypothesis Spaces

- **Complete Ignorance.** There are $2^{16} = 65536$ possible boolean functions over four input features. We can't figure out which one is correct until we've seen every possible input-output pair. After 7 examples, we still have 2^9 possibilities.

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

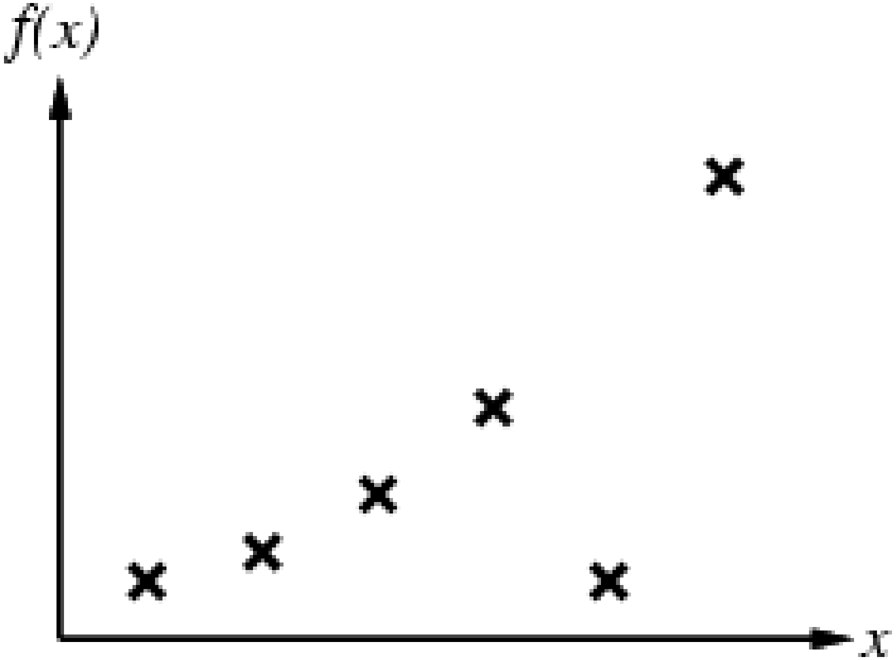
Inductive Bias

- Need to make assumptions
  - Experience alone doesn't allow us to make conclusions about unseen data instances

- Two types of bias:
  - Restriction: Limit the hypothesis space (e.g., naïve Bayes)
  - Preference: Impose ordering on hypothesis space (e.g., decision tree)

Inductive learning example

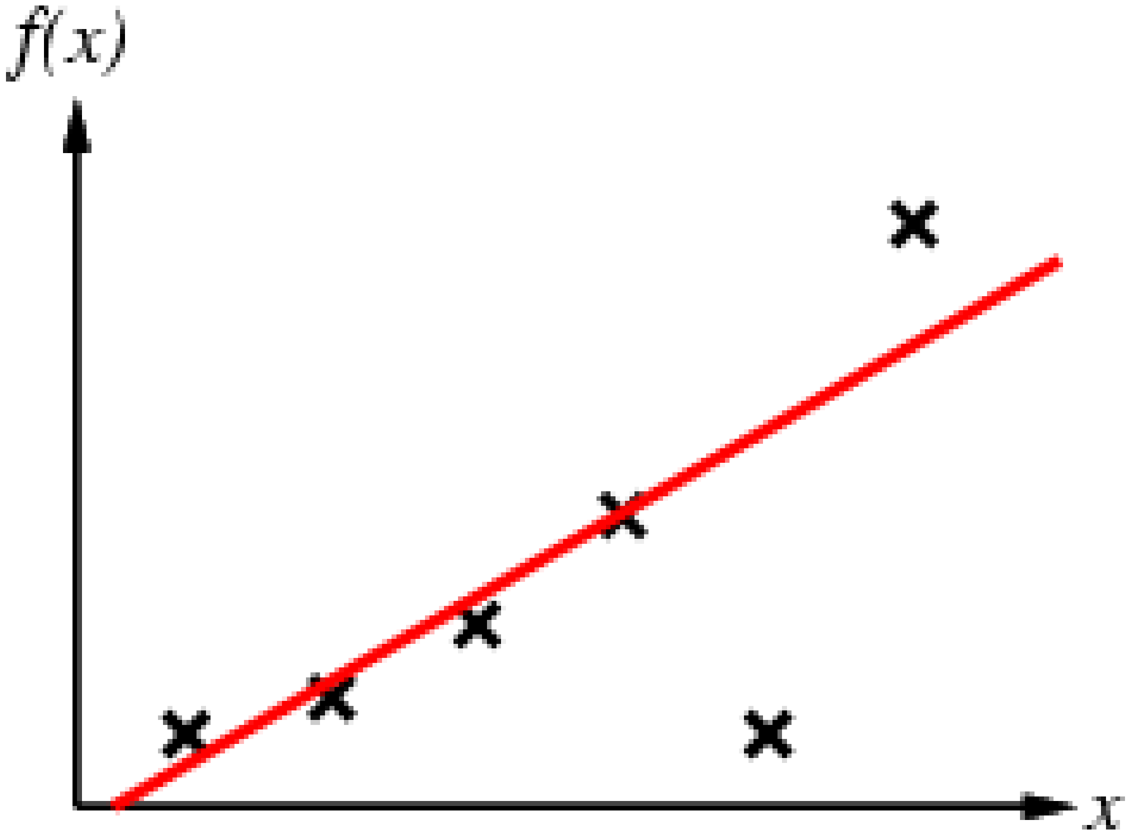
- Construct *h* to agree with *f* on training set
  - *h* is consistent if it agrees with *f* on all training examples
- E.g., curve fitting (regression):



x = Input data point (training example)

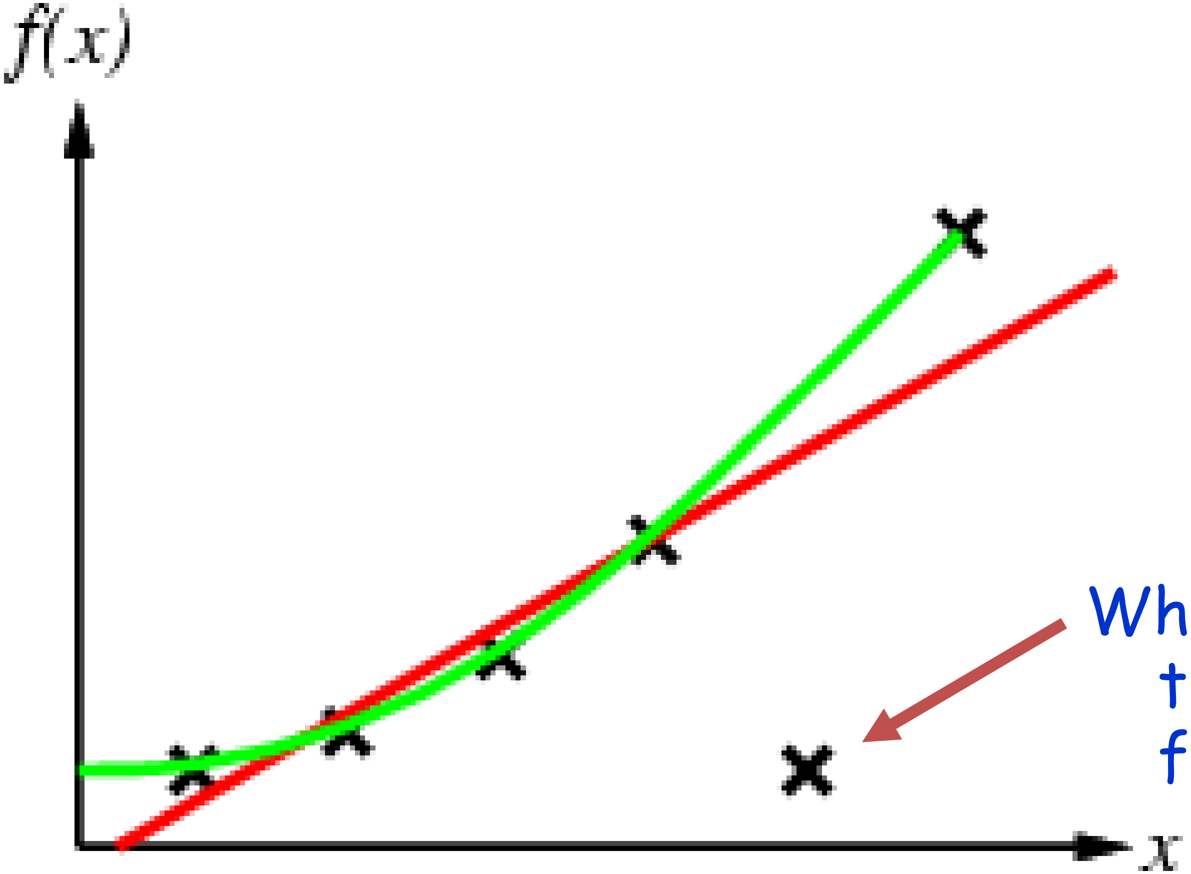
Inductive learning example

- *h* = Straight line?



Inductive learning example

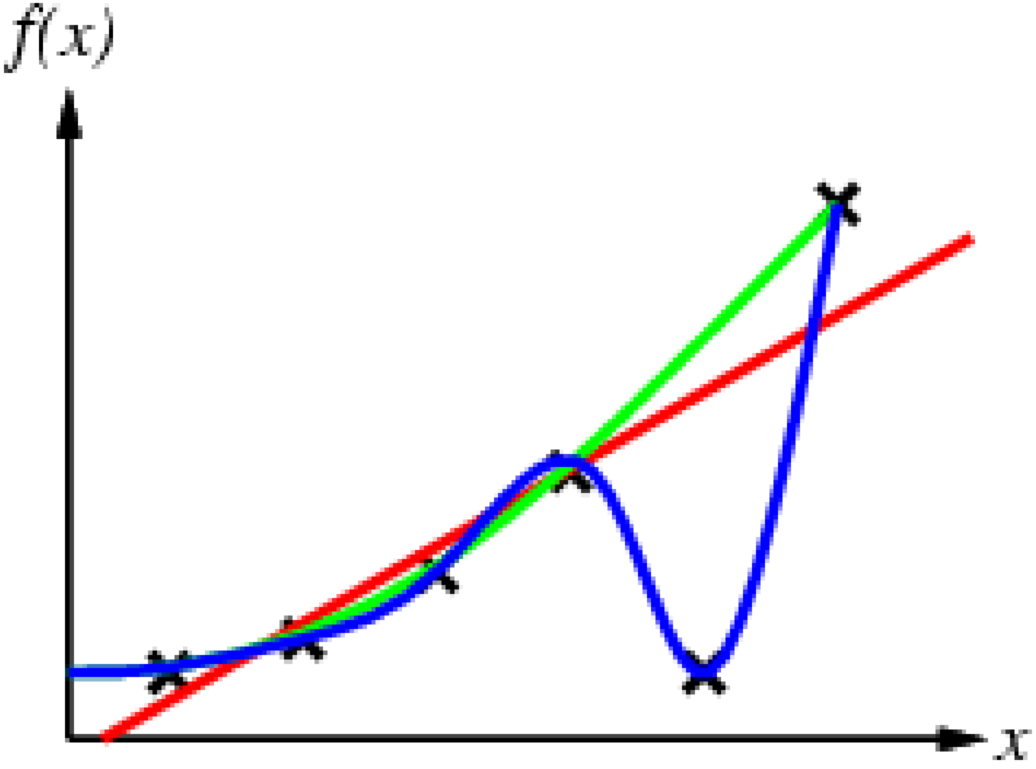
- What about a quadratic function?



What about this little fella?

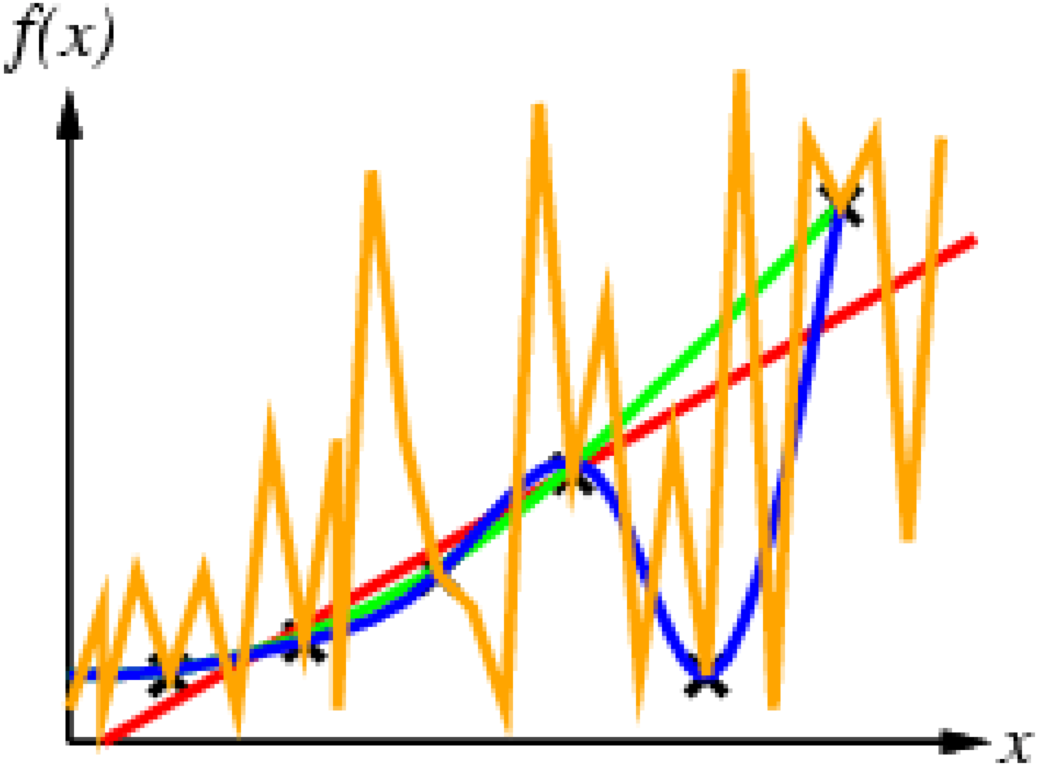
Inductive learning example

Finally, a function that satisfies all!

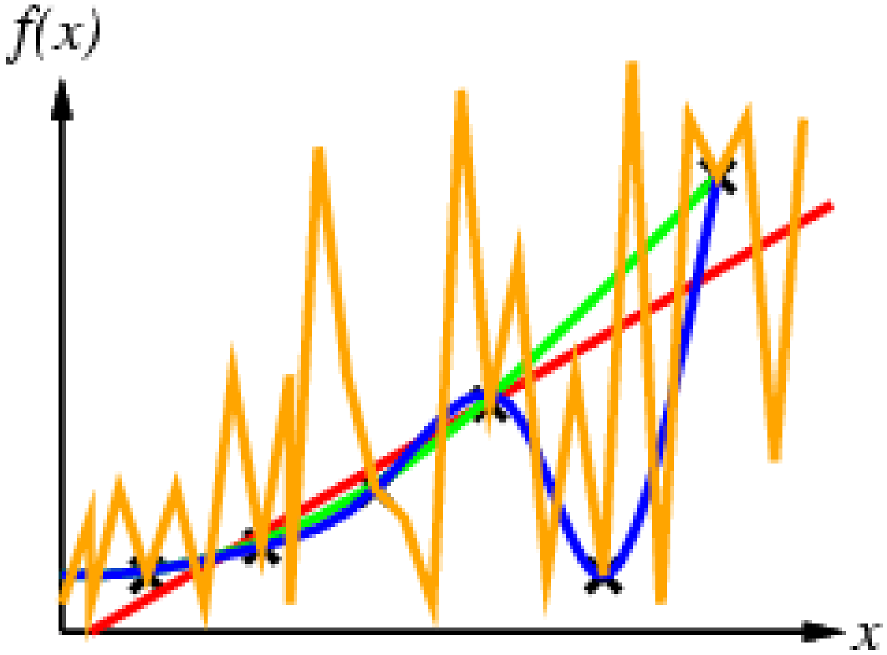


Inductive learning example

- But so does this one…



Ockham’s Razor Principle



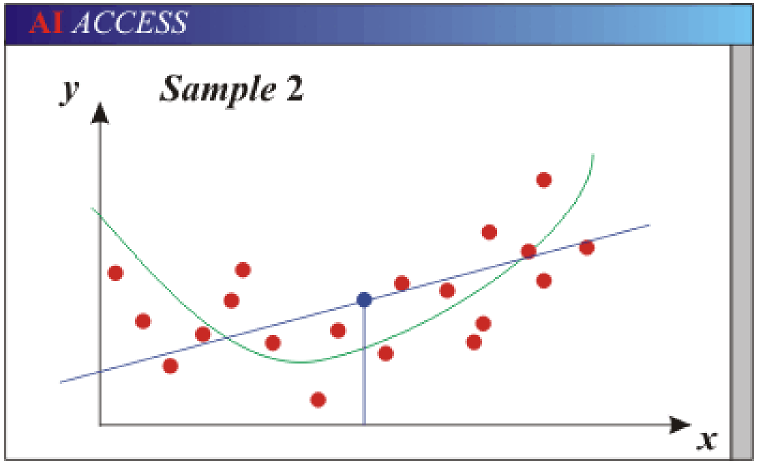
Ockham’s razor: prefer the simplest hypothesis consistent with data

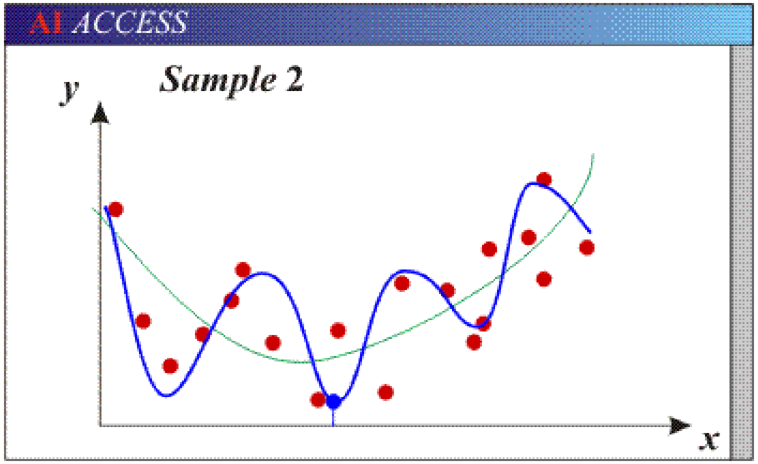
Related to KISS principle (“keep it simple stupid”)

Smooth blue function preferable over wiggly yellow one

If noise known to exist in this data, even linear might be better (the lowest x might be due to noise)

Bias-Variance Trade-off





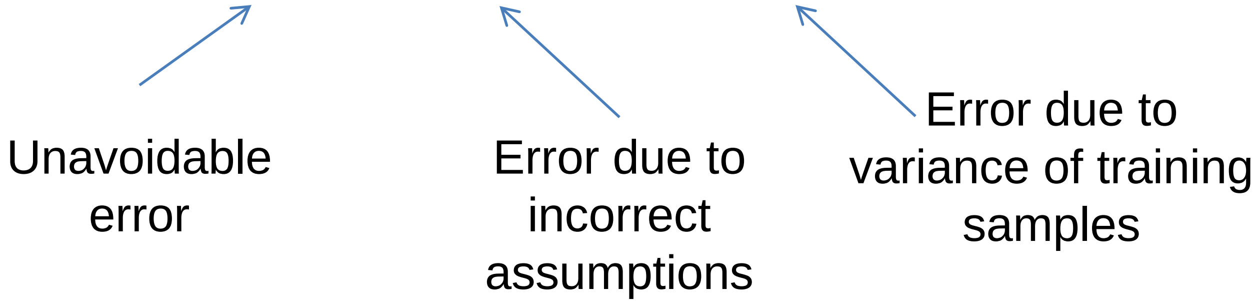
- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).

- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Slide credit: D. Hoiem

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

See the following for explanations of bias-variance (also Bishop's "Neural Networks" book):

- http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

Slide credit: D. Hoiem

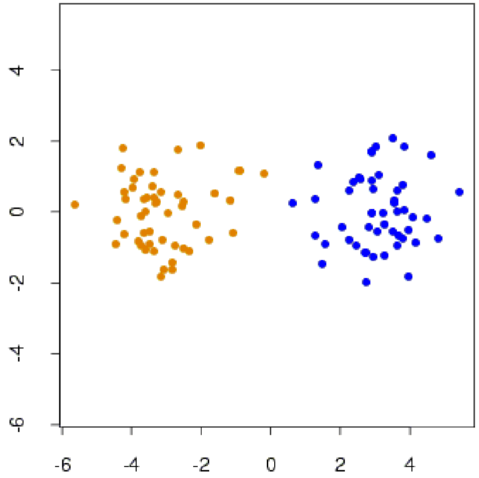
Clustering

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights

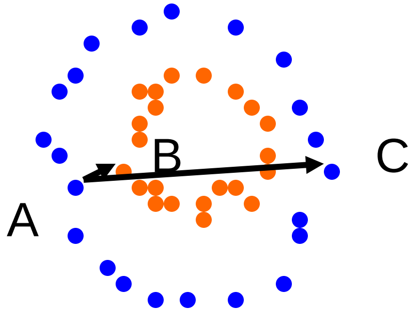
As we go down this chart, the clustering strategies have more tendency to transitively group points even if they are not nearby in feature space

Cluster Analysis

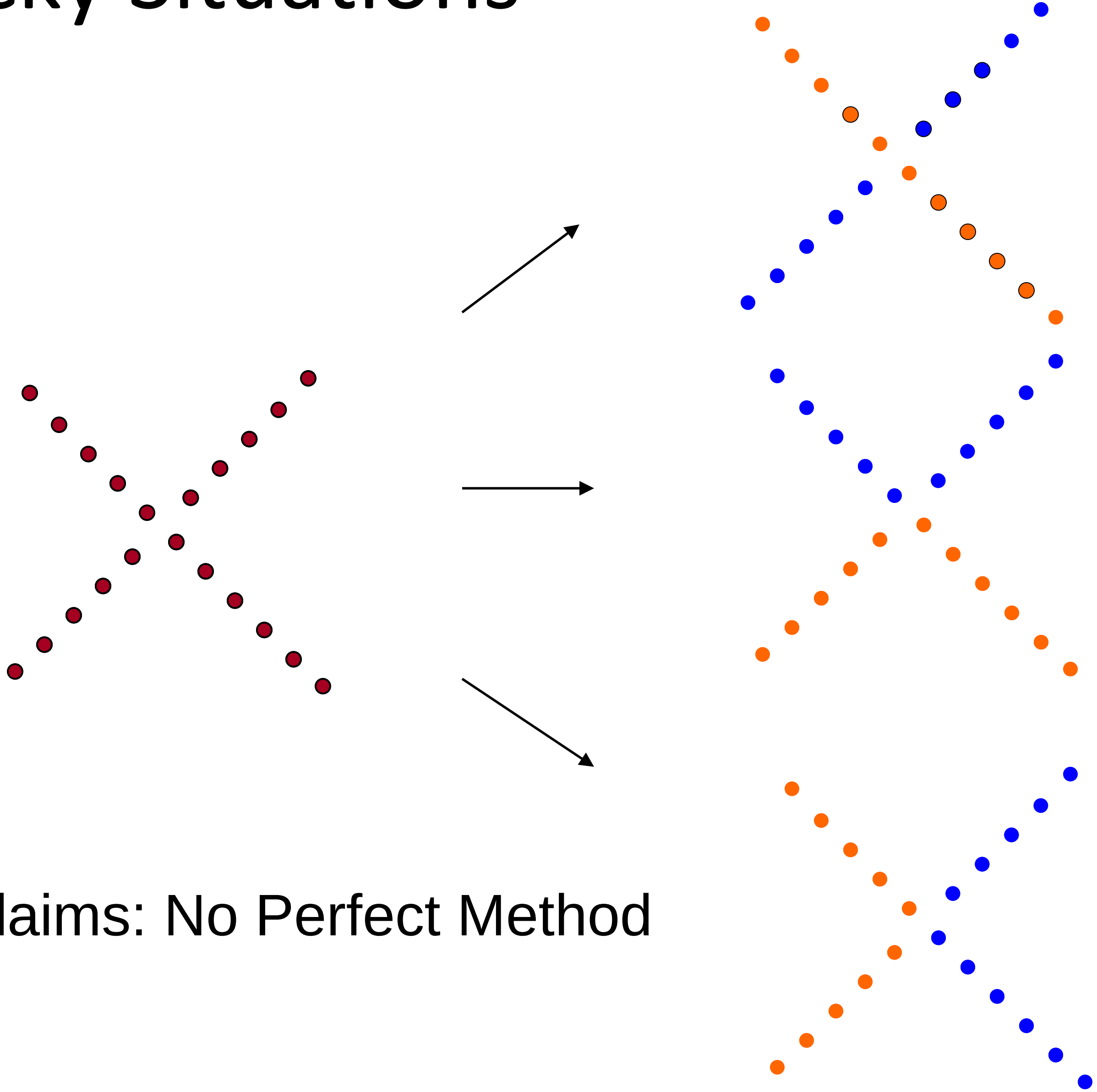
- Grouping a collection of objects into clusters, such that those within each cluster are **more closely related**
- Core problem: distance definition
  - Intra cluster distance
  - Inter cluster distance



But what if



Tricky Situations



Claims: No Perfect Method

K-Means

- Squared Euclidean Distance

$$\mathbf{d}(\mathbf{x},\mathbf{y}) = \left\|\mathbf{x}-\mathbf{y}\right\|^2$$

- Sum of Squared Error Distance

$$J = \sum_{k=1}^{K} \sum_{C(i)=k} \left\|\mathbf{x}_i - \underline{\bar{\mathbf{x}}_k}\right\|^2$$

$$\bar{\mathbf{x}}_k = \frac{1}{n_i} \sum_{C(i)=k} \mathbf{x}_i$$

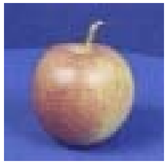
K-Means Iterative Optimization

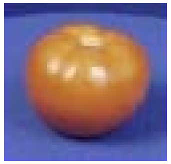
- Initialize: Randomly select k points (or partition the data into k initial clusters)
- Step 1: Compute the mean of each cluster
- Step 2: Assign each point to the closest partition
- Step 3: If any point changed its cluster membership

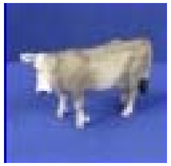
  Then repeat Step 1

Classification

- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

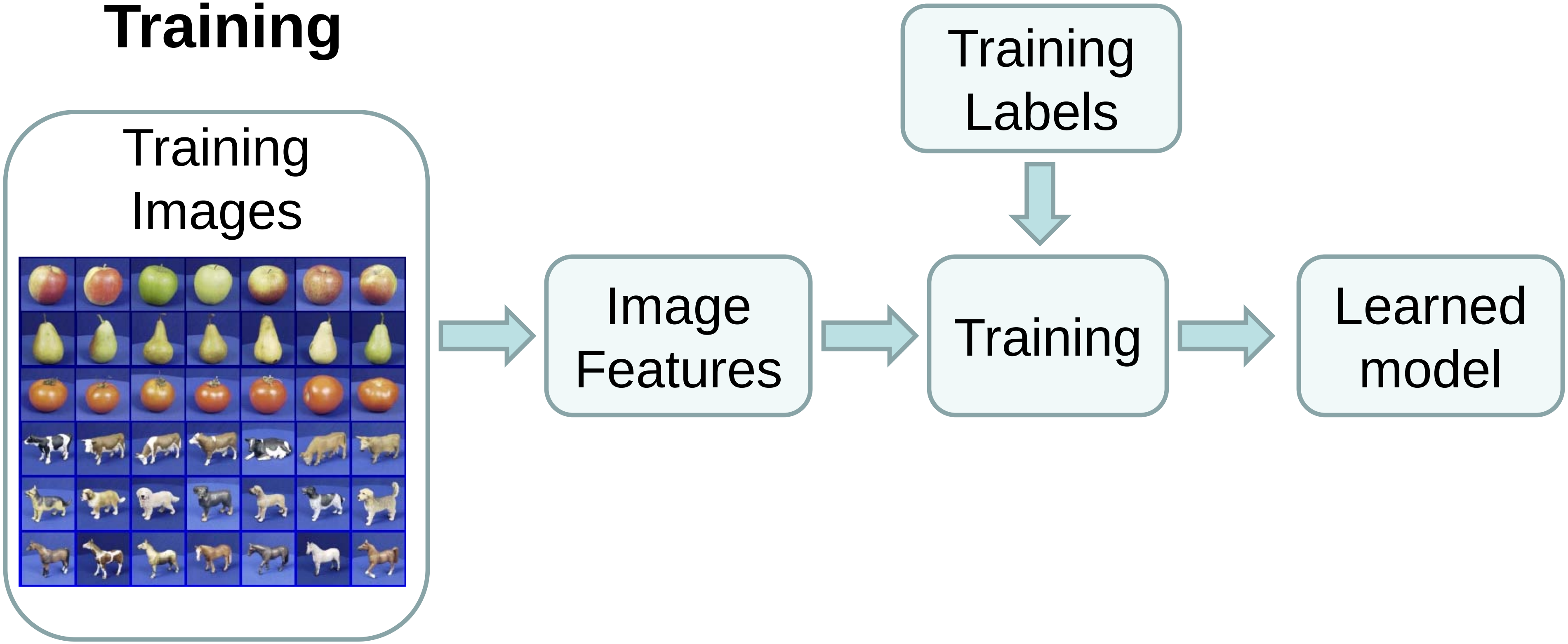
f() = “tomato”

f() = “cow”

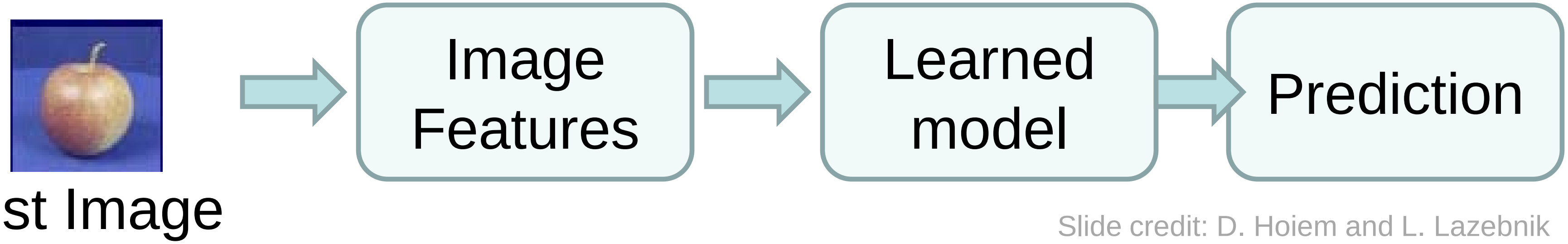
Slide credit: L. Lazebnik

Steps

Training



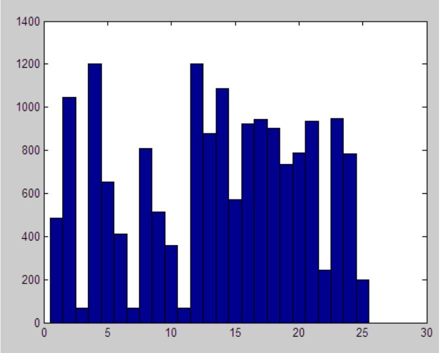
Testing

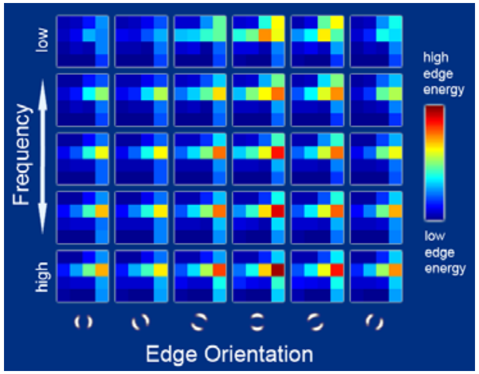


Slide credit: D. Hoiem and L. Lazebnik

Features

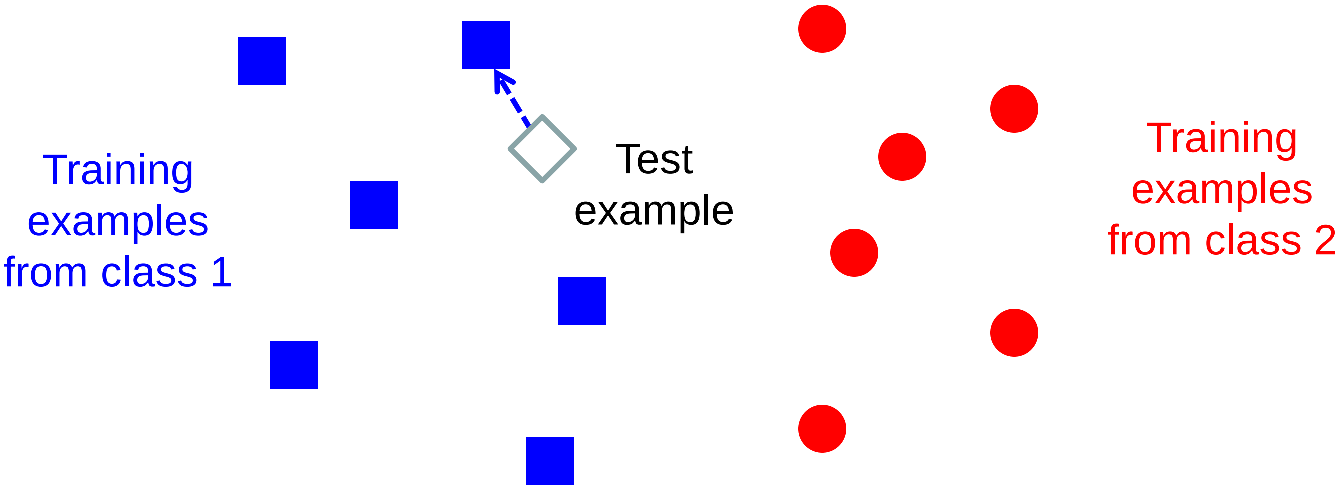
- Raw pixels
- Histograms
- GIST descriptors
- …





Slide credit: L. Lazebnik

Classifiers: Nearest neighbor

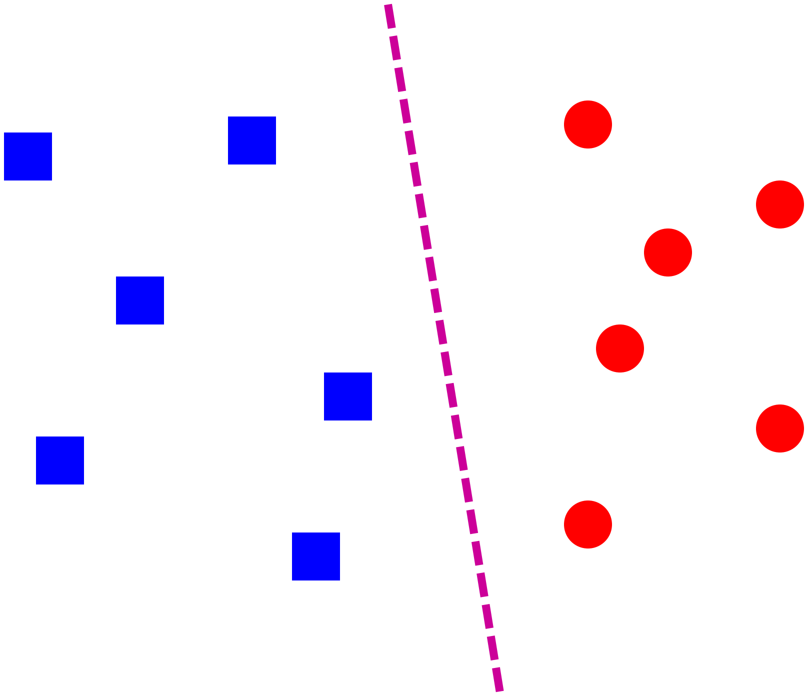


f(**x**) = label of the training example nearest to **x**

- All we need is a distance function for our inputs
- No training required!

Slide credit: L. Lazebnik

Classifiers: Linear



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Slide credit: L. Lazebnik

Many classifiers to choose from

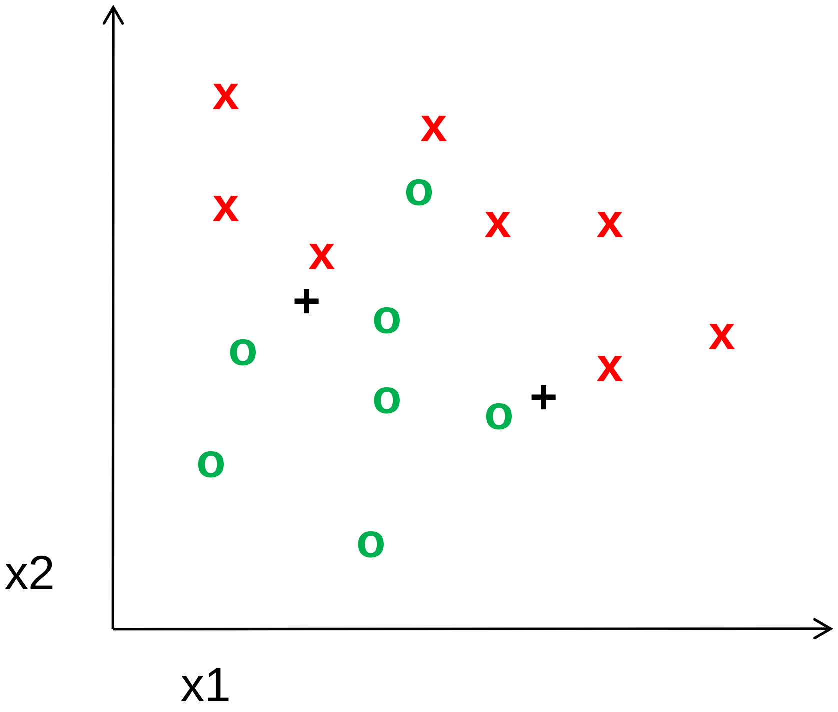
- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- Etc.

Slide credit: D. Hoiem

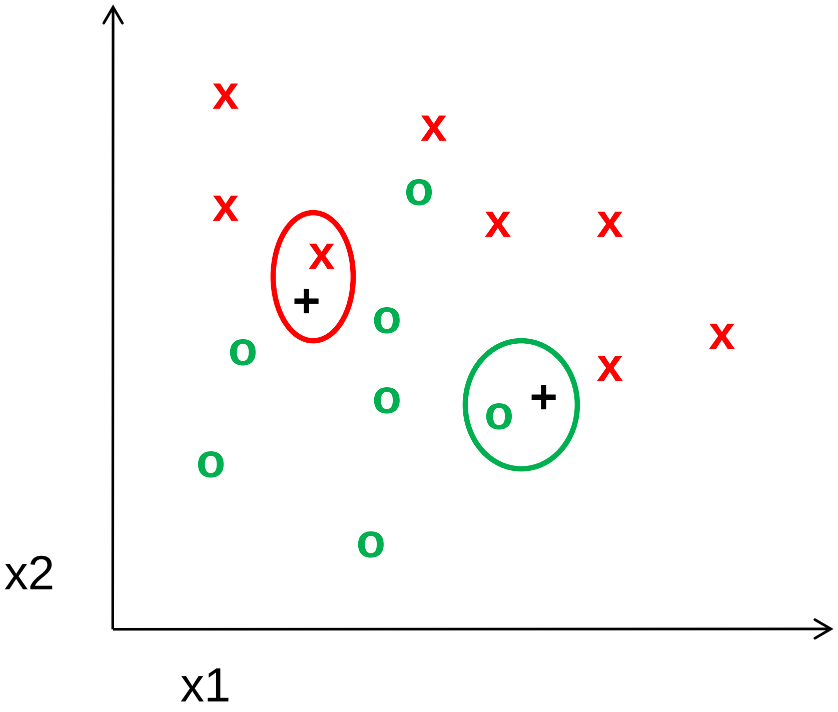
Very brief tour of some classifiers

- **K-nearest neighbor**
- **Decision Trees**
- **SVM**
- **Naïve Bayes**
- Neural networks
- Bayesian network
- **Logistic regression**
- Randomized Forests
- RBMs
- Etc.

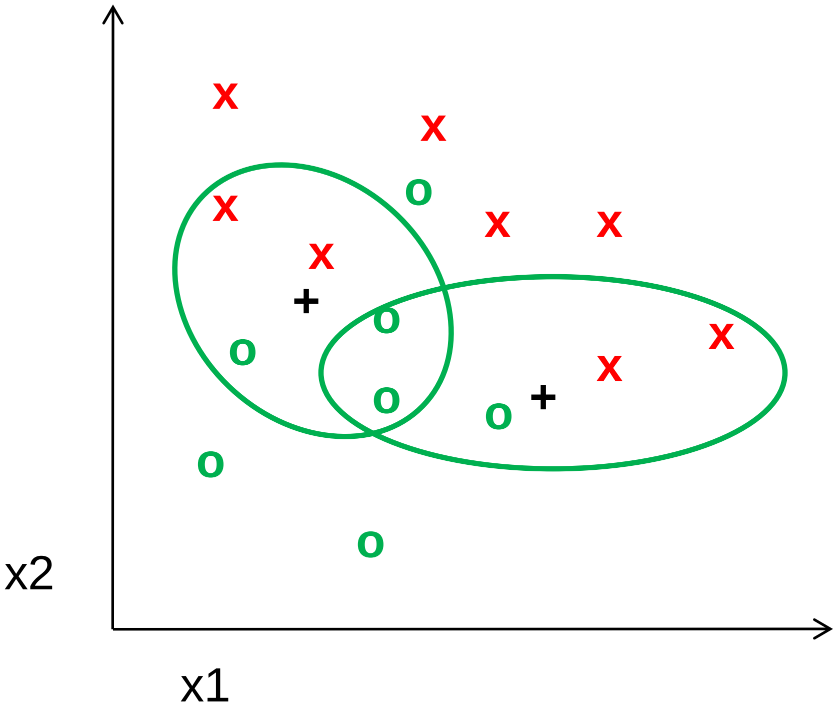
K-nearest neighbor



1-nearest neighbor

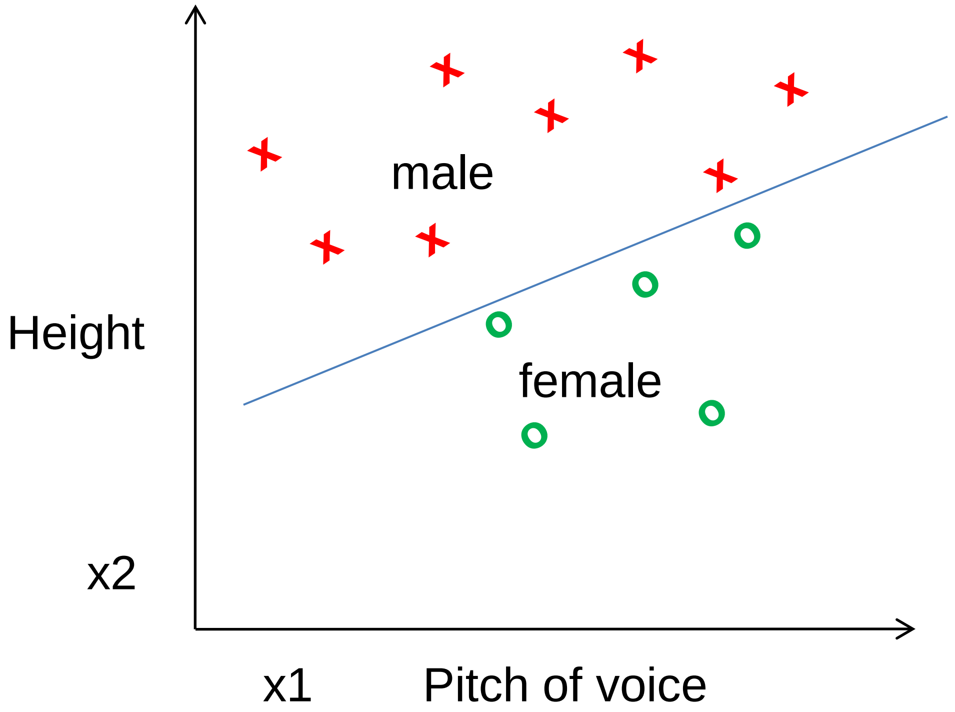


5-nearest neighbor



Classifiers: Logistic Regression

Maximize likelihood of label given data, assuming a log-linear model



$$\log \frac{P(x_1, x_2 \mid y = 1)}{P(x_1, x_2 \mid y = -1)} = \mathbf{w}^T \mathbf{x}$$

$$P(y = 1 \mid x_1, x_2) = 1/\left(1 + \exp\left(-\mathbf{w}^T \mathbf{x}\right)\right)$$

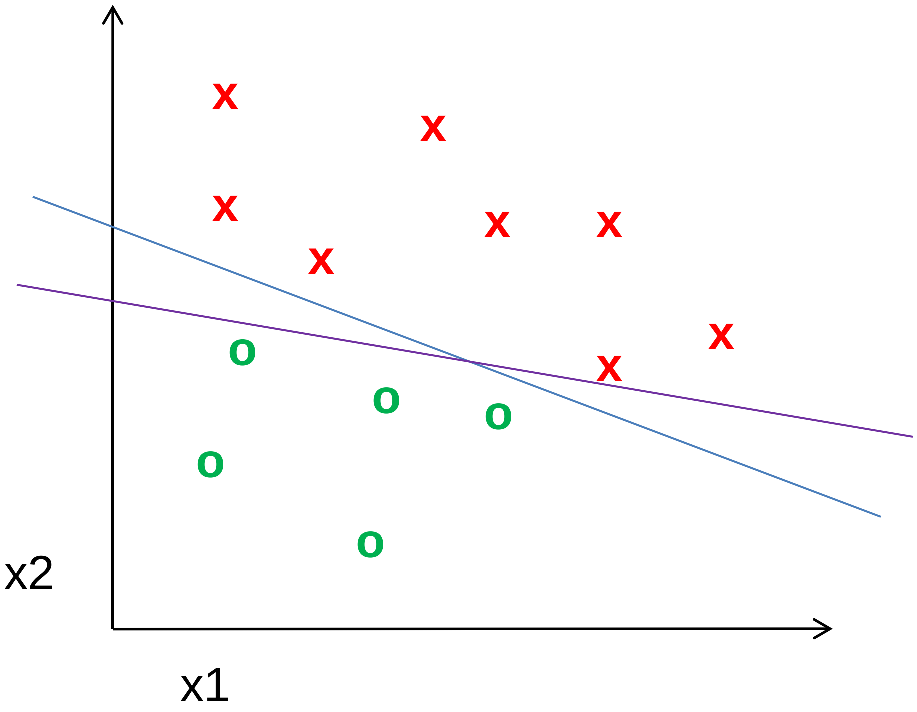
Using Logistic Regression

- Quick, simple classifier (try it first)

- Outputs a probabilistic label confidence

- Use L2 or L1 regularization
  - L1 does feature selection and is robust to irrelevant features but slower to train

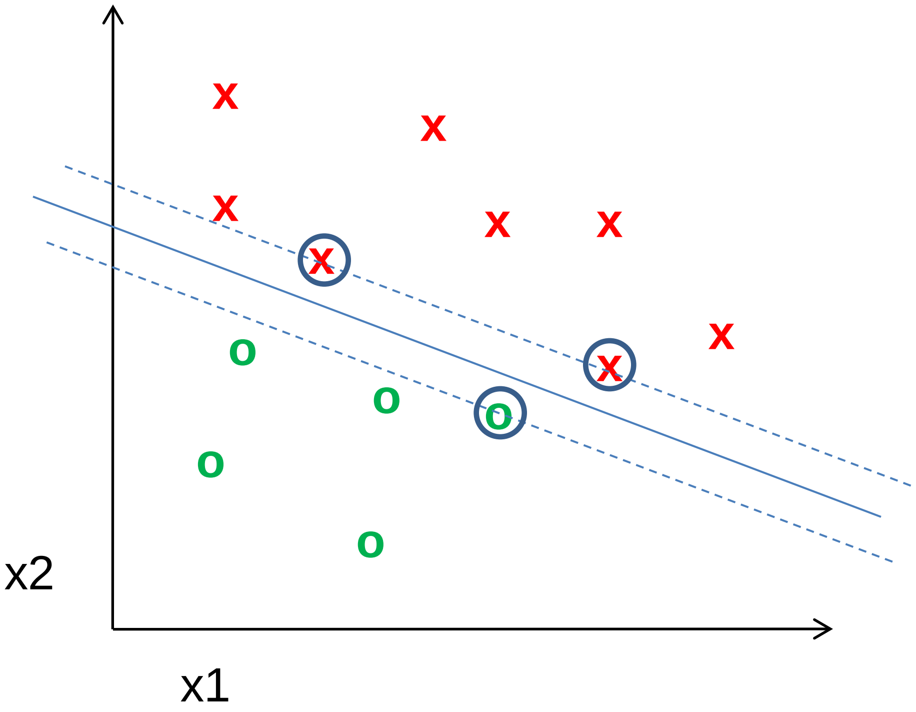
Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Classifiers: Linear SVM

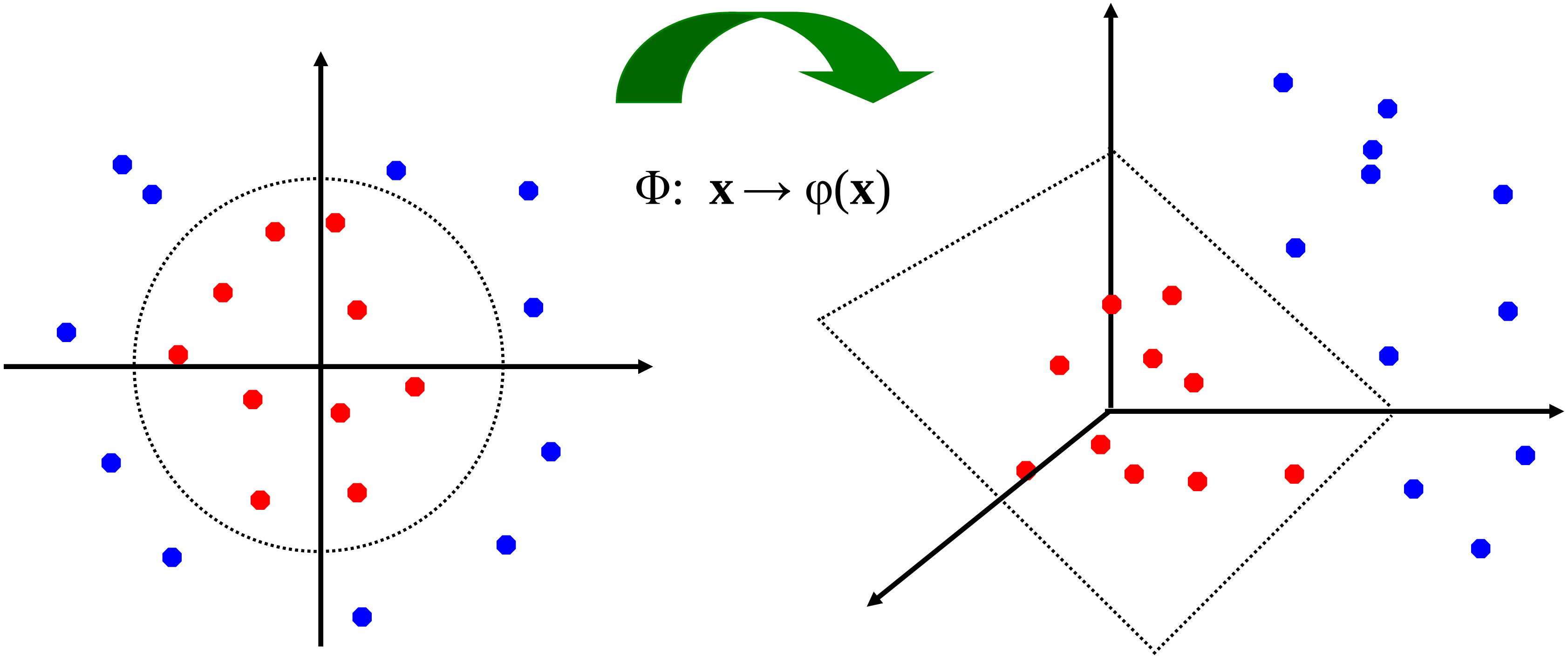


- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$

Slide credit: Andrew Moore

Nonlinear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(\mathbf{x})$, define a kernel function K such that

  $$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

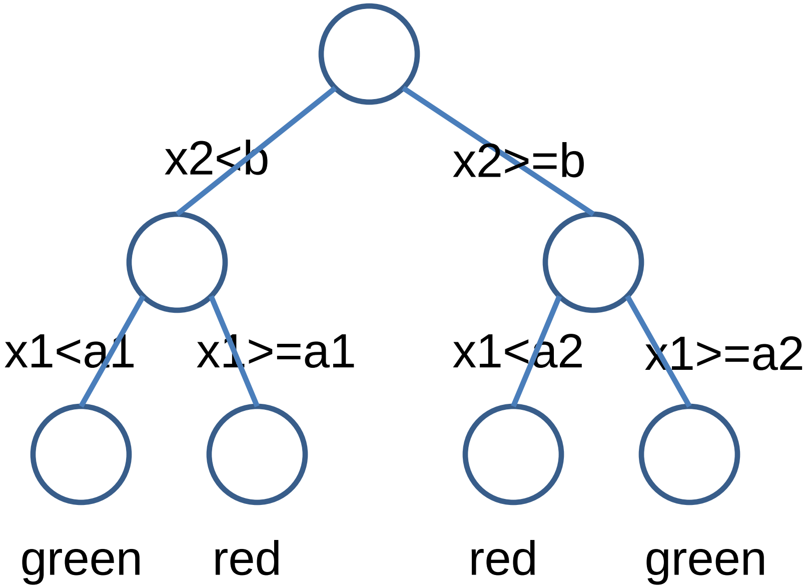
  (to be valid, the kernel function must satisfy *Mercer's condition*)

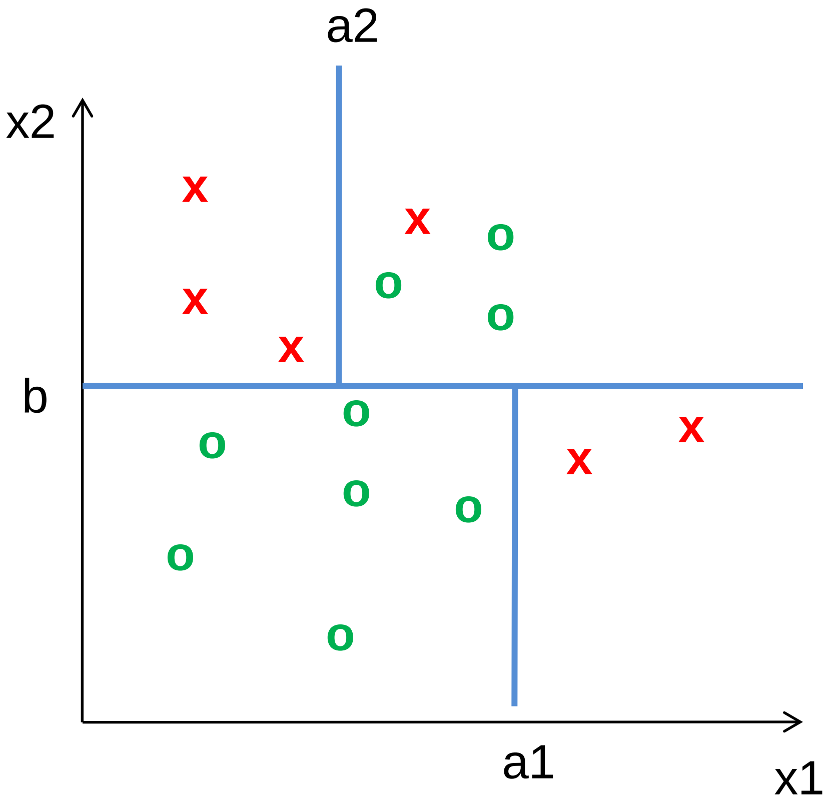
- This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}) + b = \sum_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

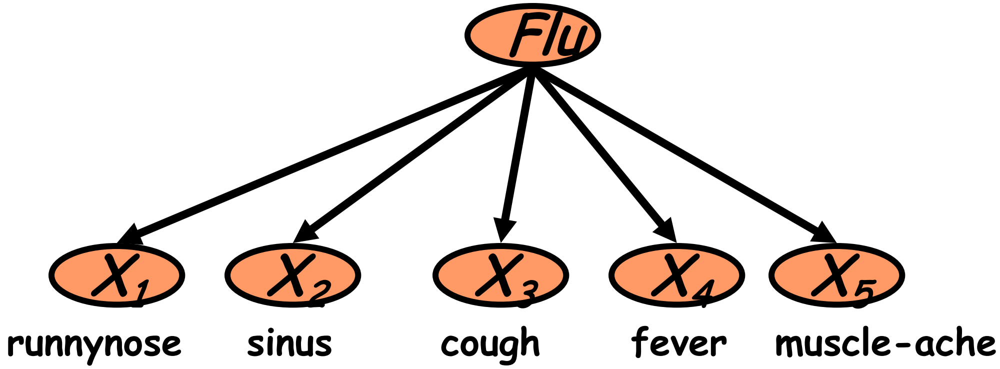
C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Decision Tree





The Naïve Bayes Classifier

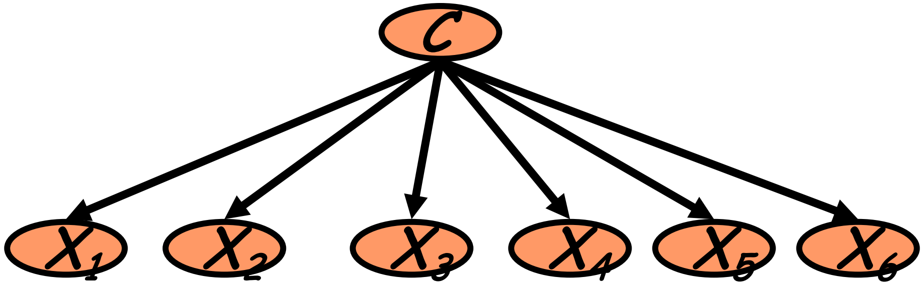


- **Conditional Independence Assumption:** features are independent of each other given the class:

$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- This model is appropriate for binary variables

Learning the Model

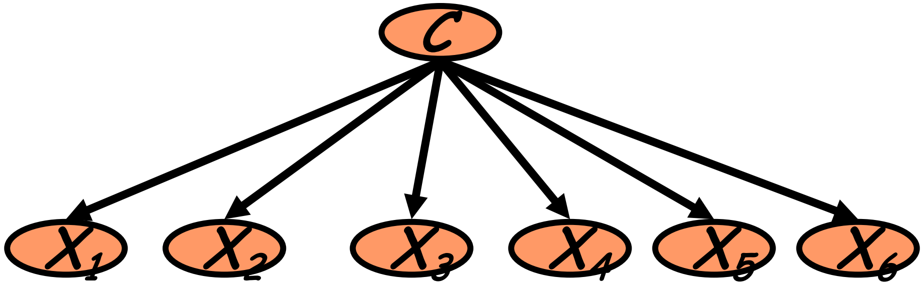


- Estimate parameters
  - simply use the frequencies in the data

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

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

Testing the Model



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

$$\sim P(X \mid C)\, P(C)$$

Using Naïve Bayes

- Simple thing to try for categorical data
- Very fast to train/test

Generative vs. Discriminative Classifiers

Generative Models

- Represent both the data and the labels
- Often, makes use of conditional independence and priors
- Examples
  - Naïve Bayes classifier
  - Bayesian network

- Models of data may apply to future prediction problems

Discriminative Models

- Learn to directly predict the labels from the data
- Often, assume a simple boundary (e.g., linear)
- Examples
  - Logistic regression
  - SVM
  - Boosted decision trees

- Often easier to predict a label from the data than to model the data

Slide credit: D. Hoiem

Two ways to think about classifiers

1. What is the objective? What are the parameters? How are the parameters learned? How is the learning regularized? How is inference performed?

2. How is the data modeled? How is similarity defined? What is the shape of the boundary?

Slide credit: D. Hoiem

What to remember about classifiers

- No free lunch: machine learning algorithms are tools, not dogmas

- Try simple classifiers first

- Better to have smart features and simple classifiers than simple features and smart classifiers

- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)

Slide credit: D. Hoiem

What else?

Where do you obtain features?

- your smartness
- feature engineering
- deep learning