MACHINE LEARNING

Undergraduate course (Spring 2020)

Nguyen Do Van, PhD

Machine learning in Robotics

- Machine Learning Introduction
- Supervised Learning
- Unsupervised Learning
- Neural Network and Deep Learning
- Reinforcement Learning

AI Related Fields

- Machine Learning.
- Data Science, Mining and Knowledge Discovery.
- Computer Vision.
- Natural Language Processing.
- Speech Recognition.
- Evolutionary and Natural Computation.
- Fuzzy Computation and Technologies.
- Artificial Life.
- Knowledge-Based Systems.
- Automated Reasoning.
- Logic and Constraint Programming.
- Intelligent Planning.
- ....

MACHINE LEARNING

Learning & Adaptation

- ”Modification of a behavioral tendency by expertise.” (Webster 1984)
- ”A learning machine, broadly defined is any device whose actions are influenced by past experiences.” (Nilsson 1965)
- ”Any change in a system that allows it to perform better the second time on repetition of the same task or on another task drawn from the same population.” (Simon 1983)
- ”An improvement in information processing ability that results from information processing activity.” (Tanimoto 1990)

Why “Learn”?

- Machine learning is programming computers to optimize a performance criterion using example data or past experience.
- There is no need to “learn” to calculate payroll
- Learning is used when:
  - Human expertise does not exist (navigating on Mars),
  - Humans are unable to explain their expertise (speech recognition)
  - Solution changes in time (routing on a computer network)
  - Solution needs to be adapted to particular cases (user biometrics)

Why "Learn"?

- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.
- Example in retail: Customer transactions to consumer behavior:

  People who bought "Da Vinci Code" also bought "The Five People You Meet in Heaven" (www.amazon.com)

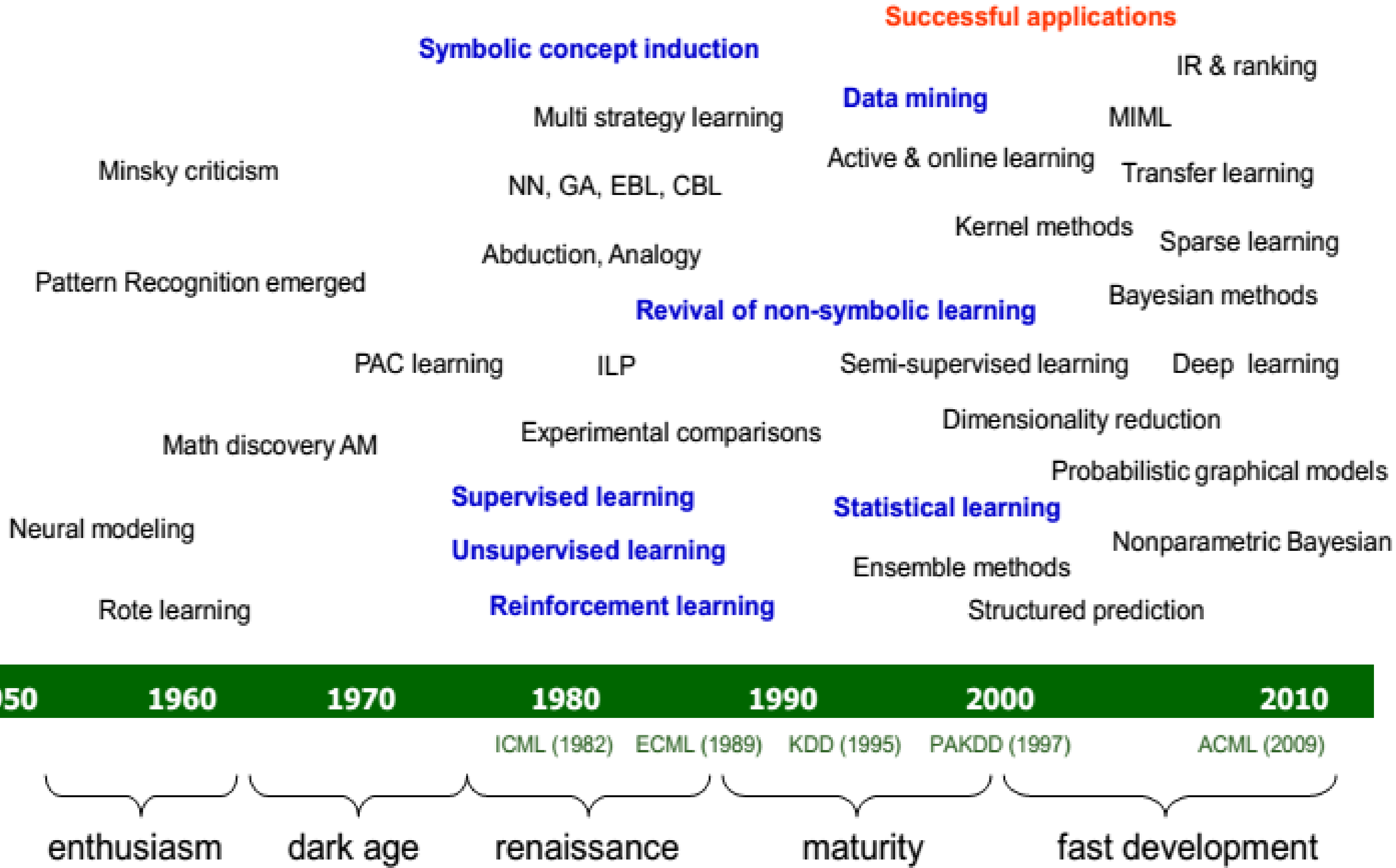
- Build a model that is *a good and useful approximation* to the data.

Learning

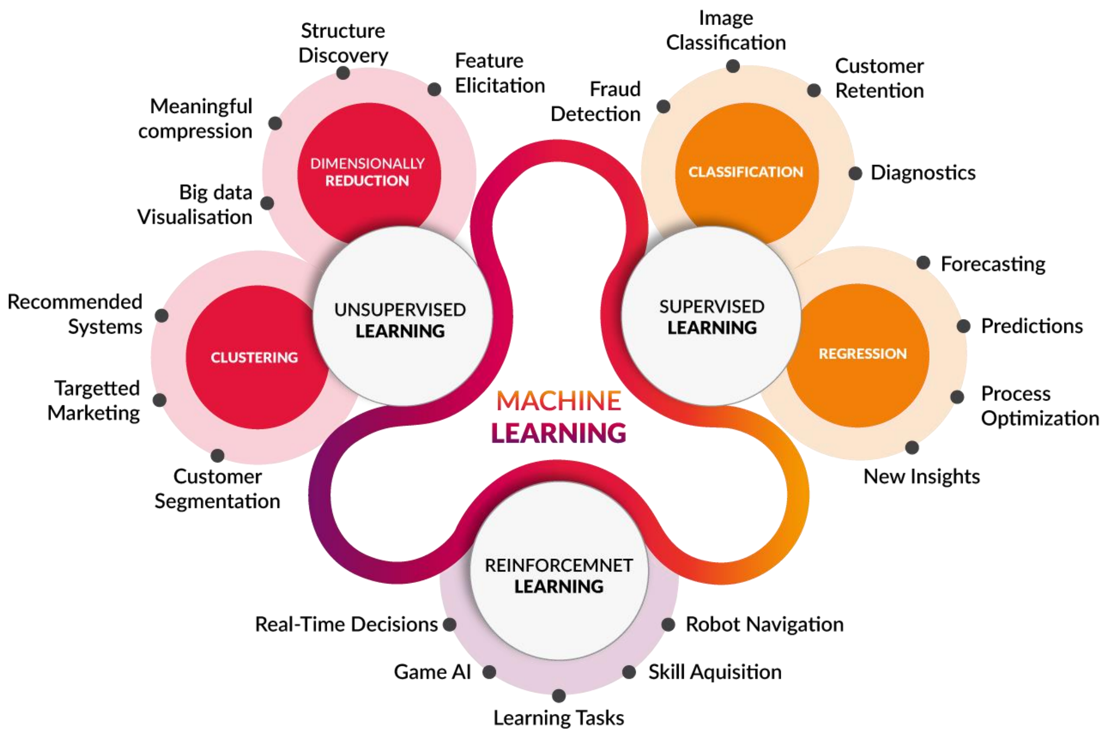
Definition:

A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience.

Machine Learning (Pre-Deep Learning)



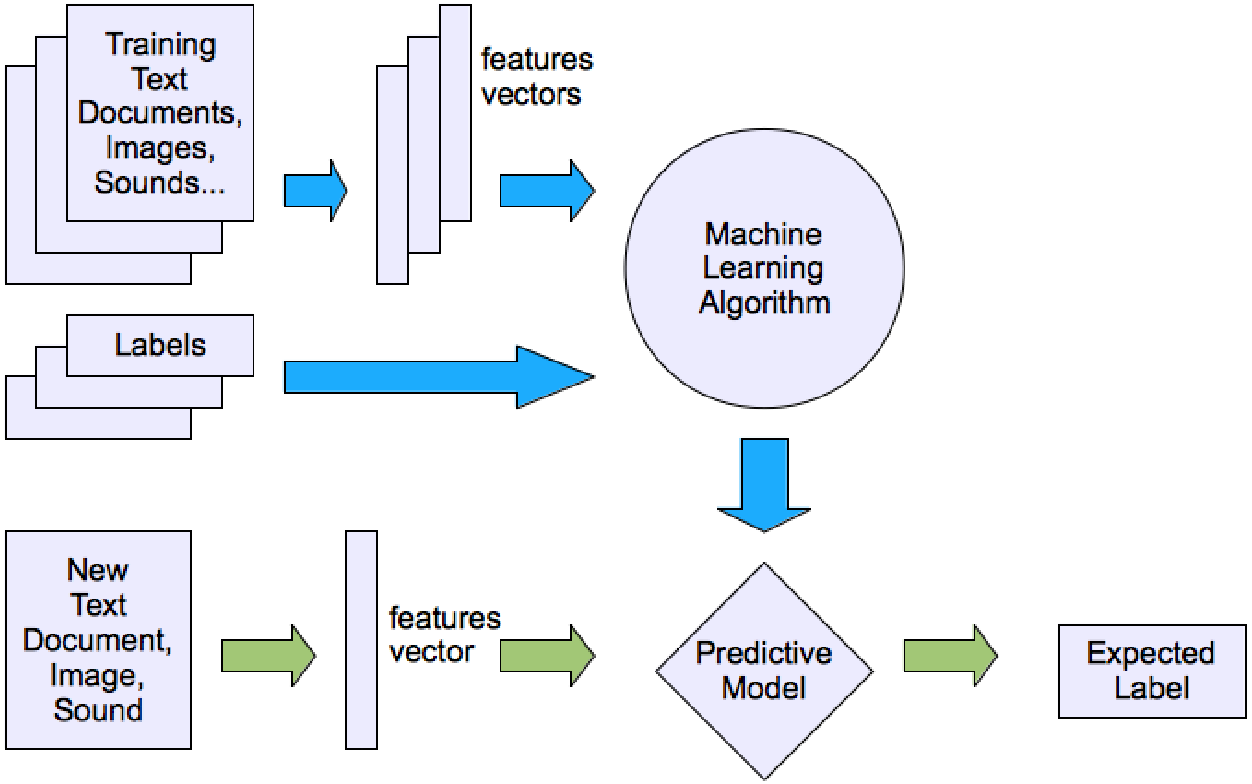
Machine Learning Methods



- Supervised learning: uses a series of labelled examples with direct feedback
- Unsupervised/clustering learning: no feedback
- Semi-supervised: using both labelled and unlabelled data
- Reinforcement learning: indirect feedback, after many examples

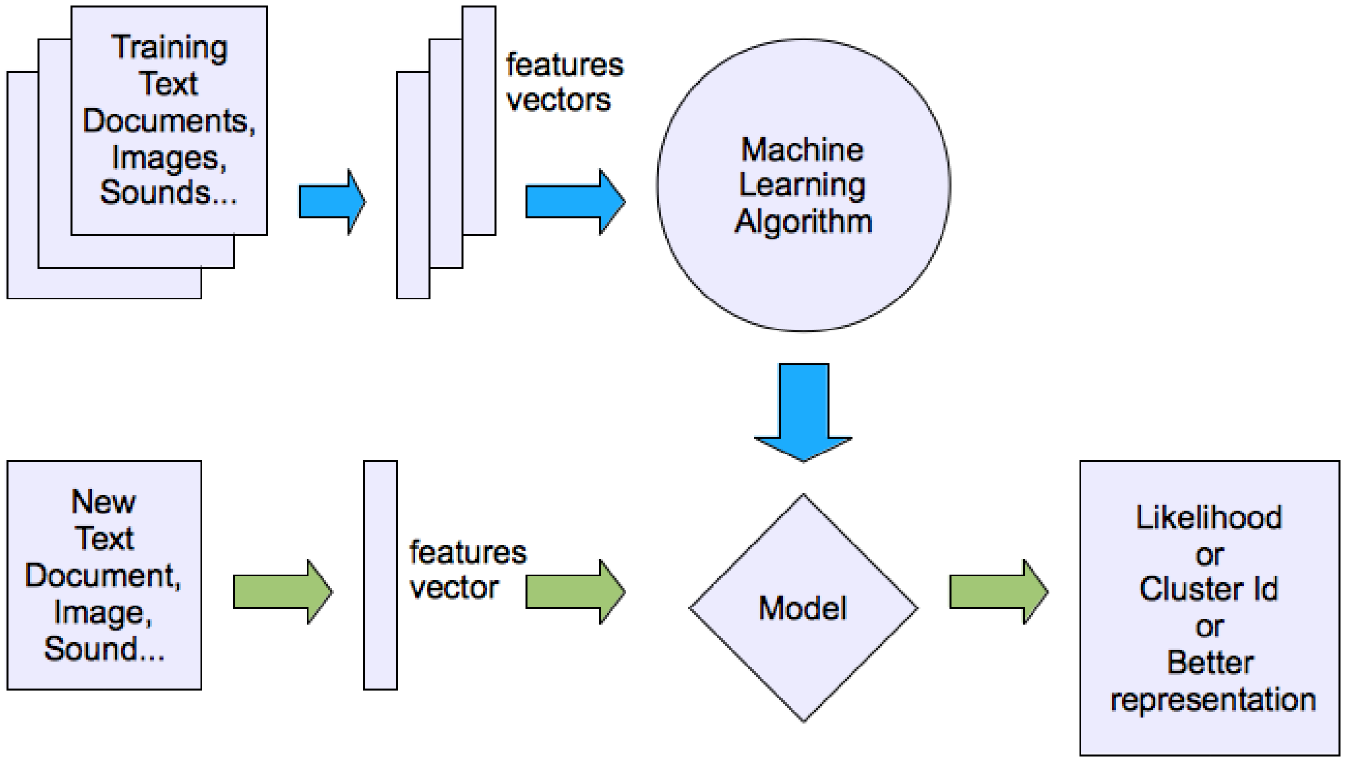
Machine learning structure

- Supervised learning

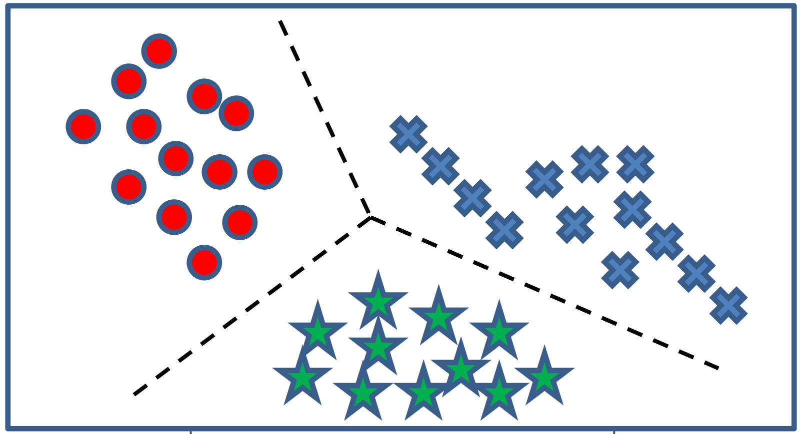


Machine learning structure

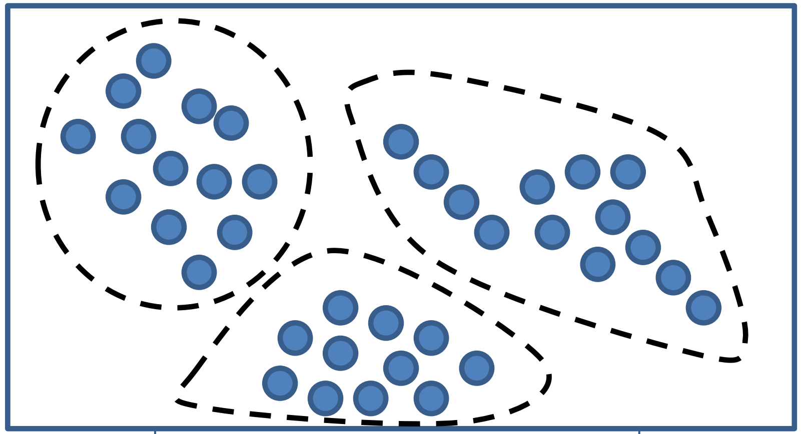
- Unsupervised learning



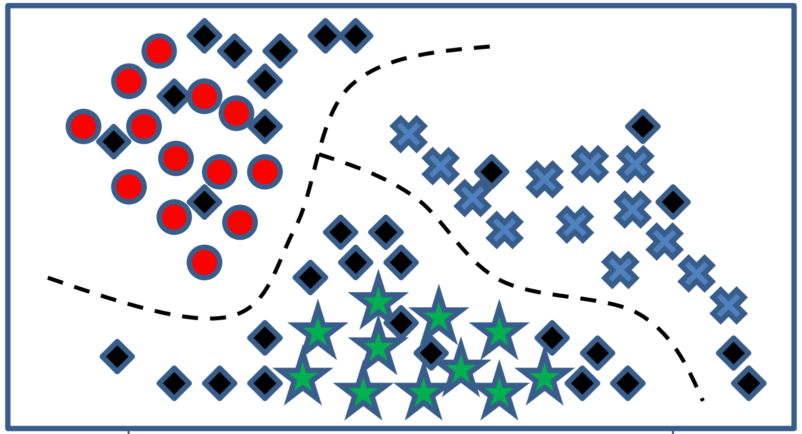
Algorithms



Supervised learning



Unsupervised learning



Semi-supervised learning

What are we seeking?

- Supervised: Low E-out or maximize probabilistic terms

$$\boxed{error = \frac{1}{N}\sum_{n=1}^{N}[y_n \neq g(x_n)]}$$

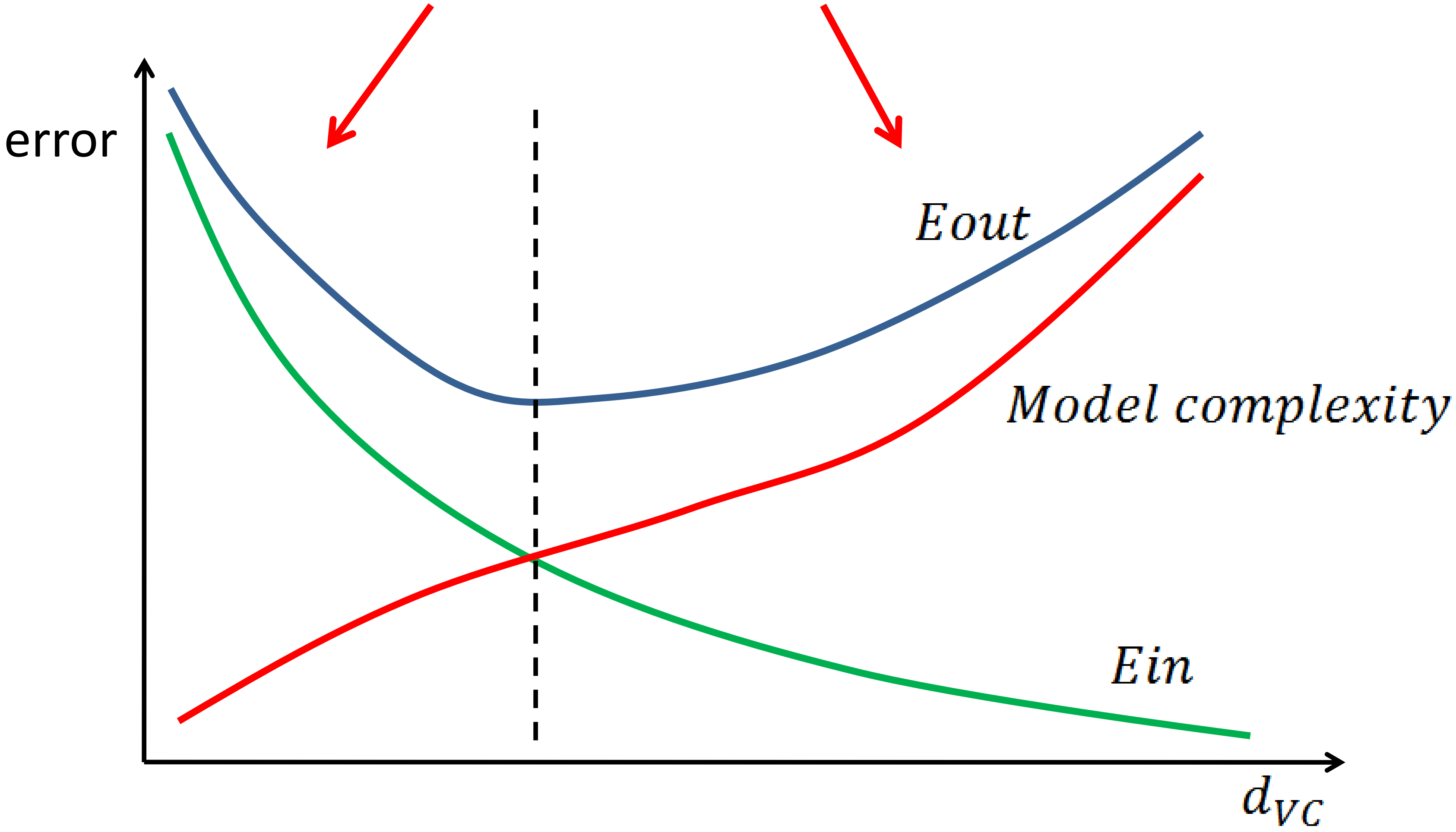
E-in: for training set
E-out: for testing set

$$\boxed{Eout(g) \leq Ein(g) \pm O\left(\sqrt{\frac{d_{VC}}{N} ln\, N}\right)}$$

- Unsupervised: Minimum quantization error, Minimum distance, MAP, MLE(maximum likelihood estimation)

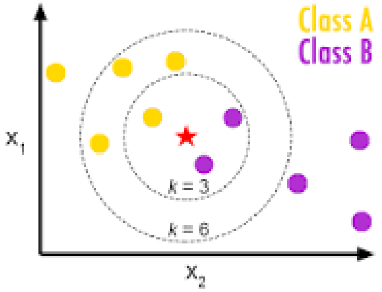
What are we seeking?

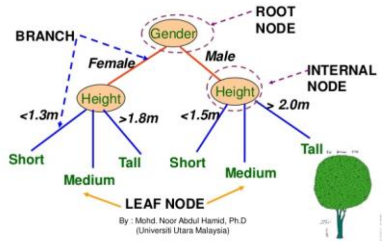
Under-fitting VS. Over-fitting (fixed *N*)

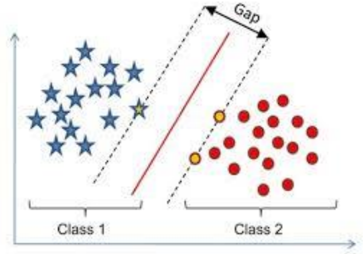


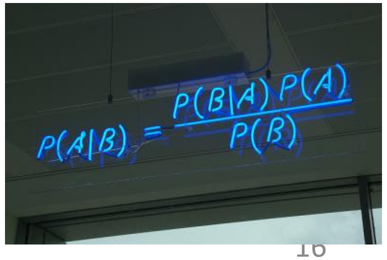
Learning techniques

- Supervised learning categories and techniques
  - **Linear classifier** (numerical functions)
  - **Parametric** (Probabilistic functions)
    - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
  - **Non-parametric** (Instance-based functions)
    - *K*-nearest neighbors, Kernel regression, Kernel density estimation, Local regression
  - **Non-metric** (Symbolic functions)
    - Classification and regression tree (CART), decision tree
  - **Aggregation**
    - Bagging (bootstrap + aggregation), Adaboost, Random



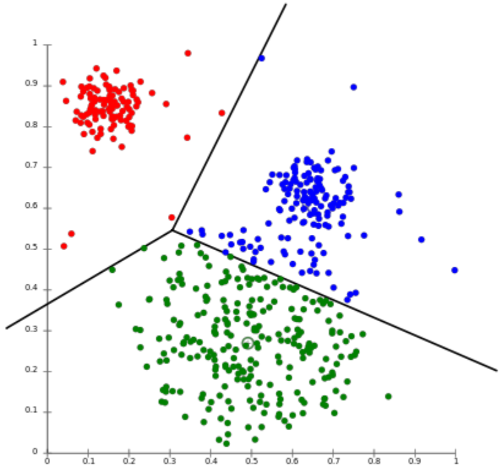


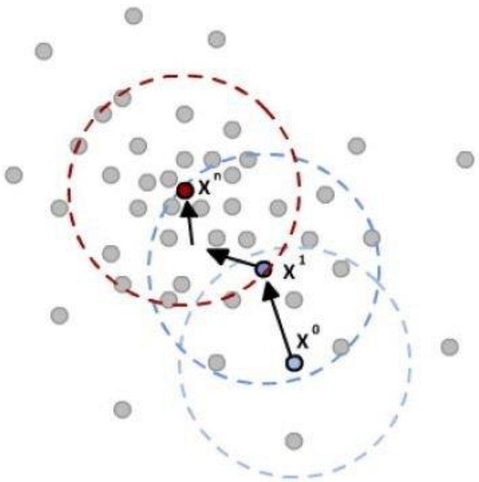


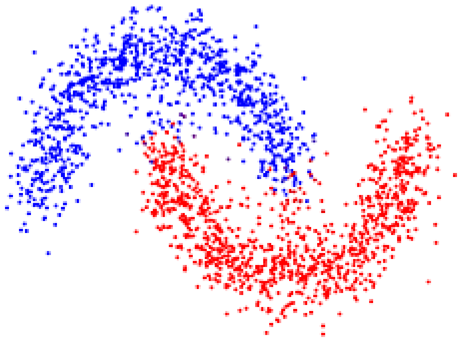


Learning Techniques

- Unsupervised Learning
  - K-means
  - Mean-shift
  - Spectral Clustering

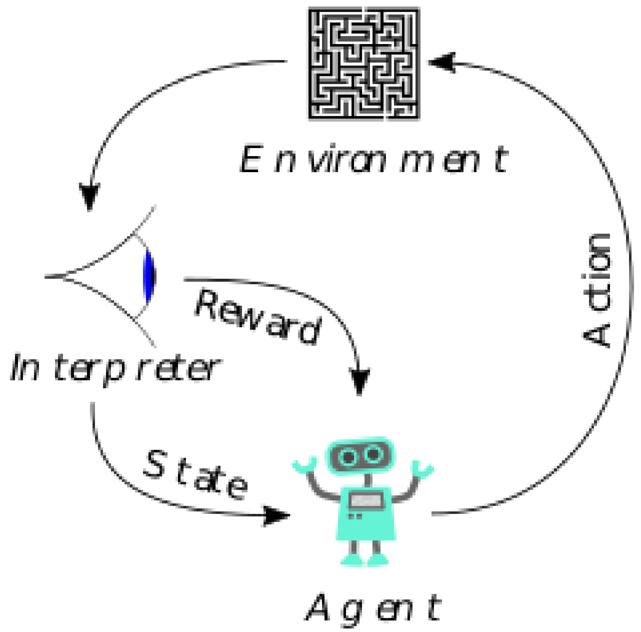




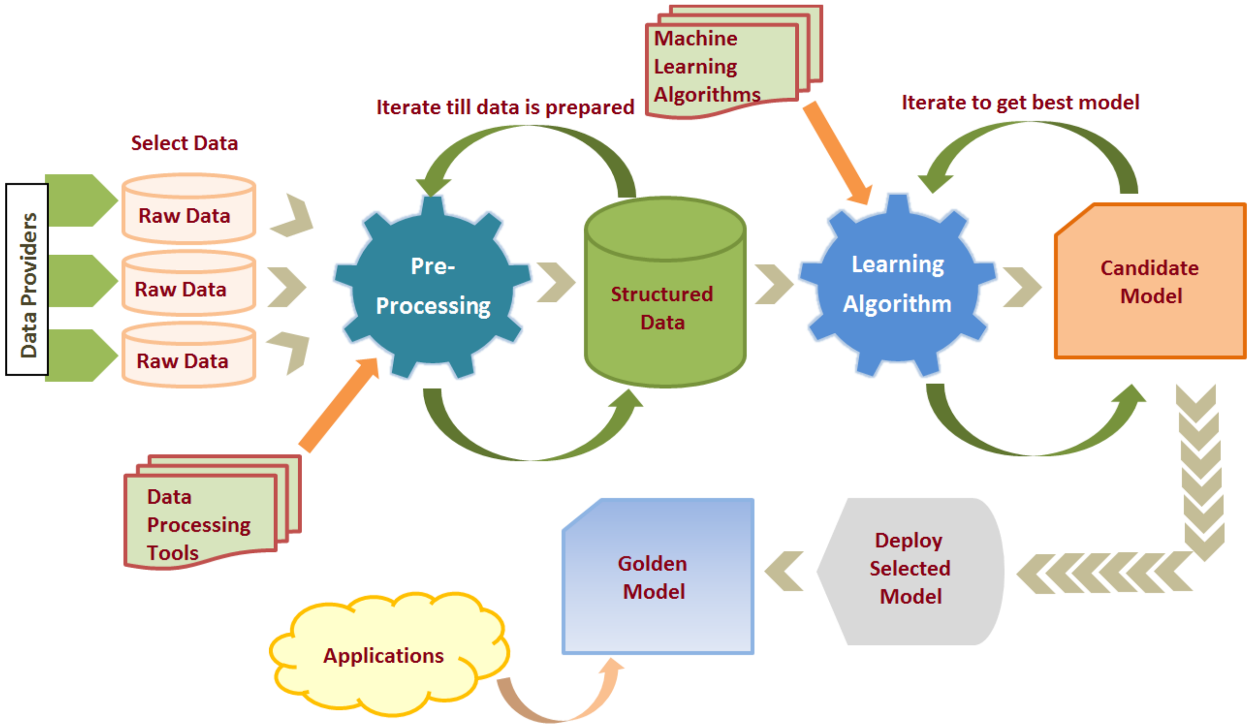


Learning Techniques

- Reinforcement learning
  - Making good decision to do new task: fundamental challenge in Artificial Intelligence and Machine learning
  - Learn to make good sequence of decisions
  - Intelligent agents learning and acting
    - Learning by trial-and-error, in real time
    - Improve with experience
    - Inspired by psychology:
      - Agents + environment
      - Agents select action to maximize *cumulative* rewards



Machine Learning Pipeline



HUMAN MACHINE INTERACTION LABORATORY

SUPERVISED LEARNING

The machine learning framework

- 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

The machine learning framework

$$\mathbf{y = f(x)}$$

output

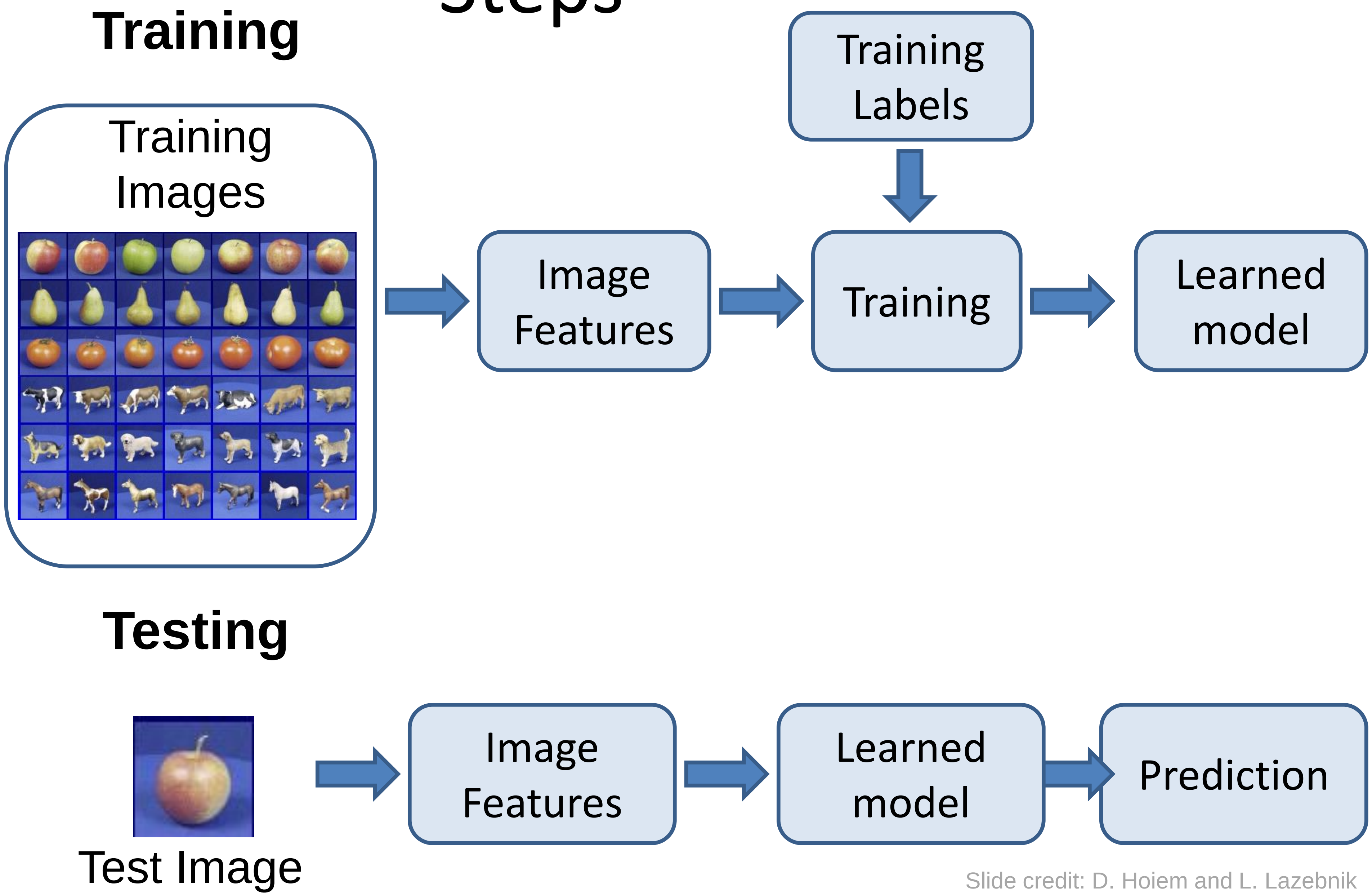
prediction function

Image feature

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Slide credit: L. Lazebnik

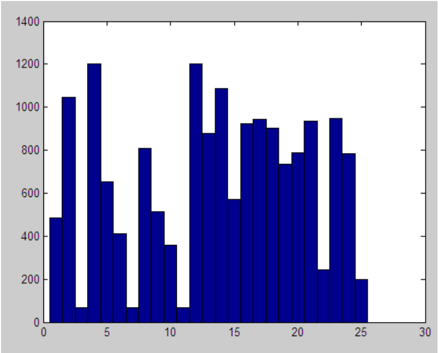
Steps

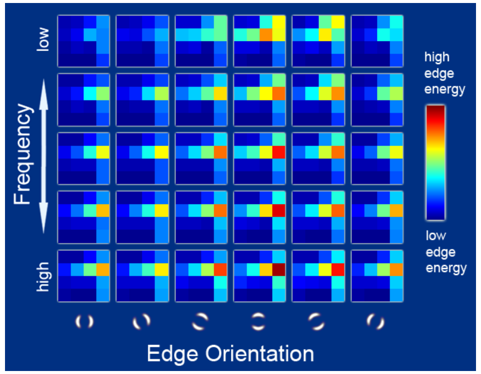


Slide credit: D. Hoiem and L. Lazebnik

Features

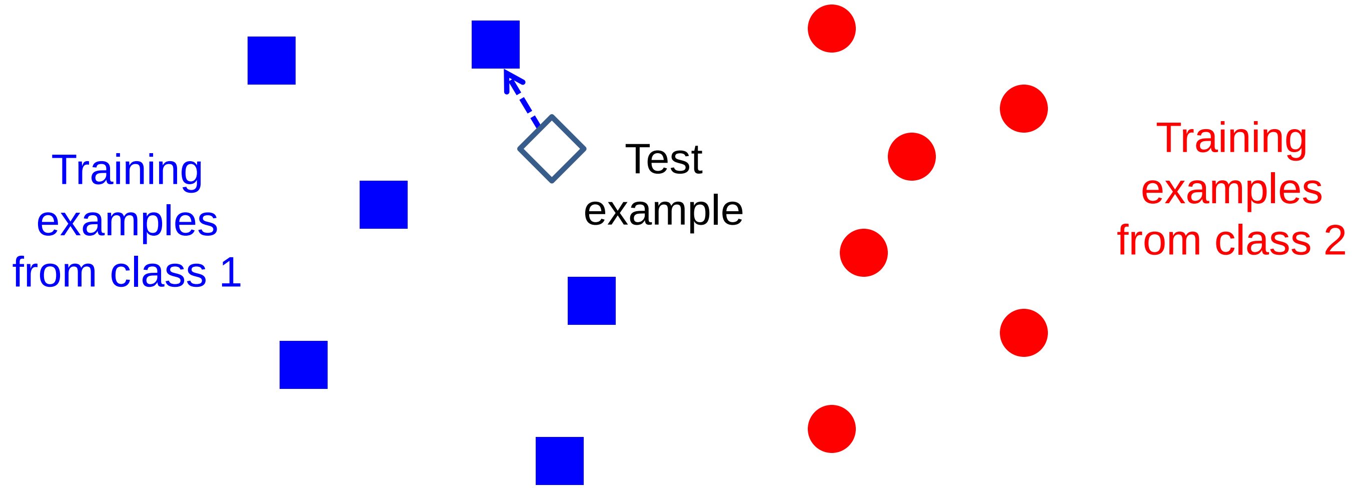
- Raw pixels
- Histograms
- GIST descriptors
- …

Slide credit: L. Lazebnik

Classifiers: Nearest neighbor

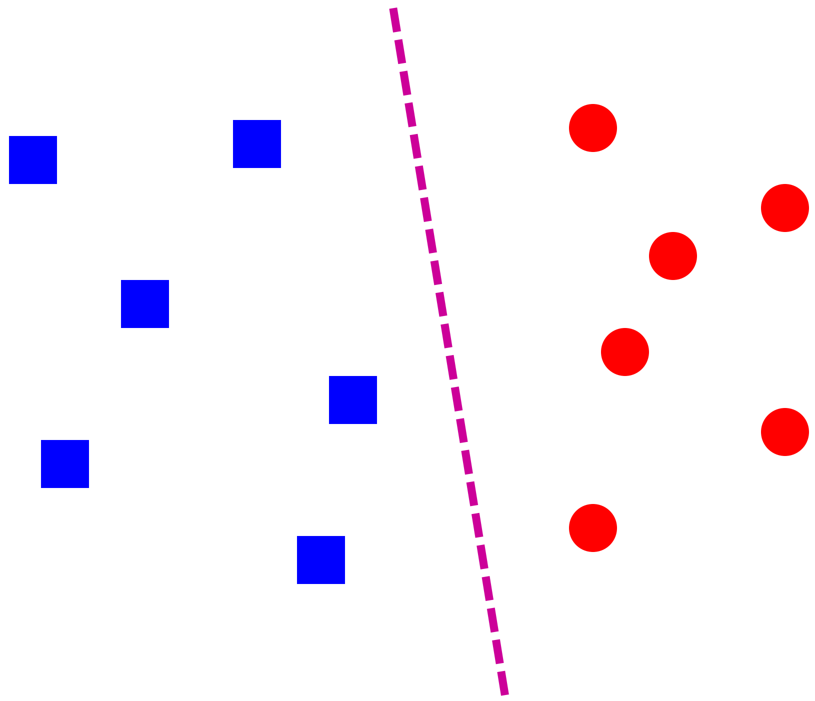


$f(\mathbf{x})$ = label of the training example nearest to $\mathbf{x}$

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

Slide credit: L. Lazebnik

Classifiers: Linear



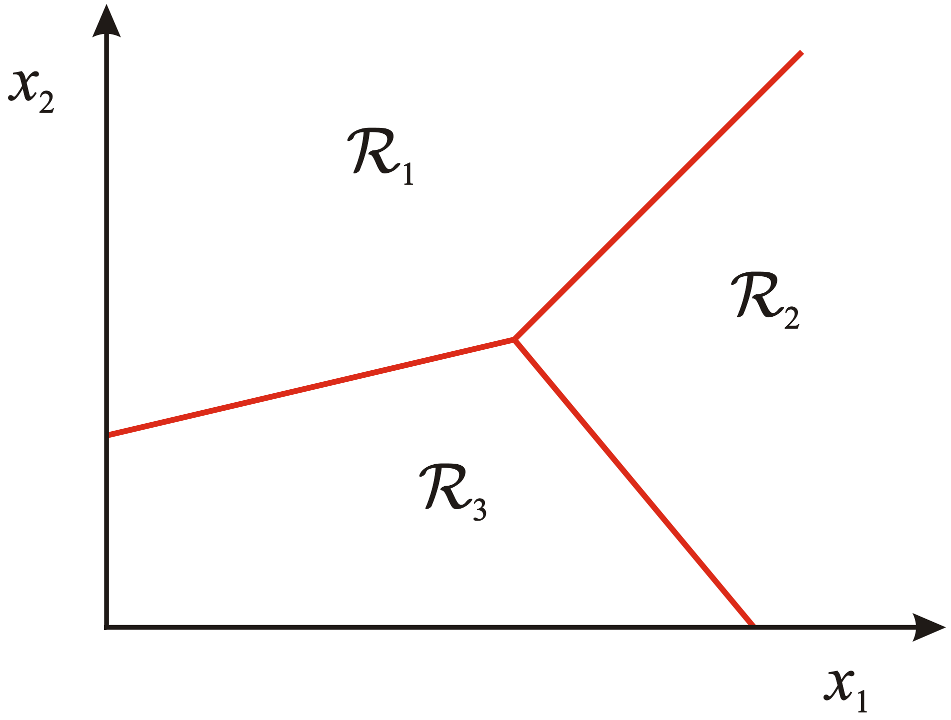
- Find a *linear function* to separate the classes:

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

Slide credit: L. Lazebnik

Classification

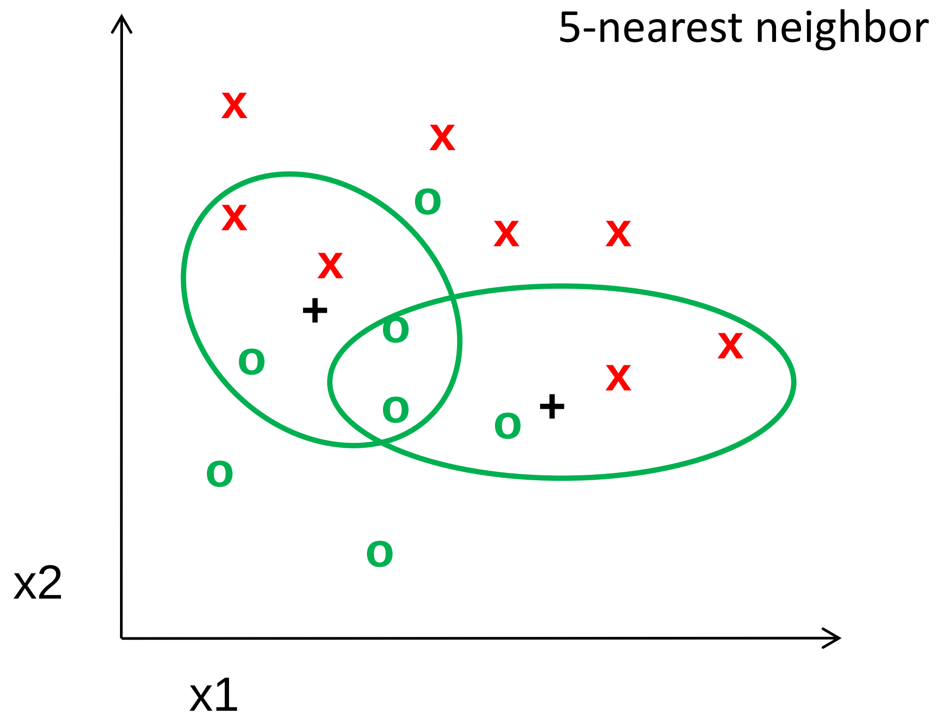
- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



Slide credit: L. Lazebnik

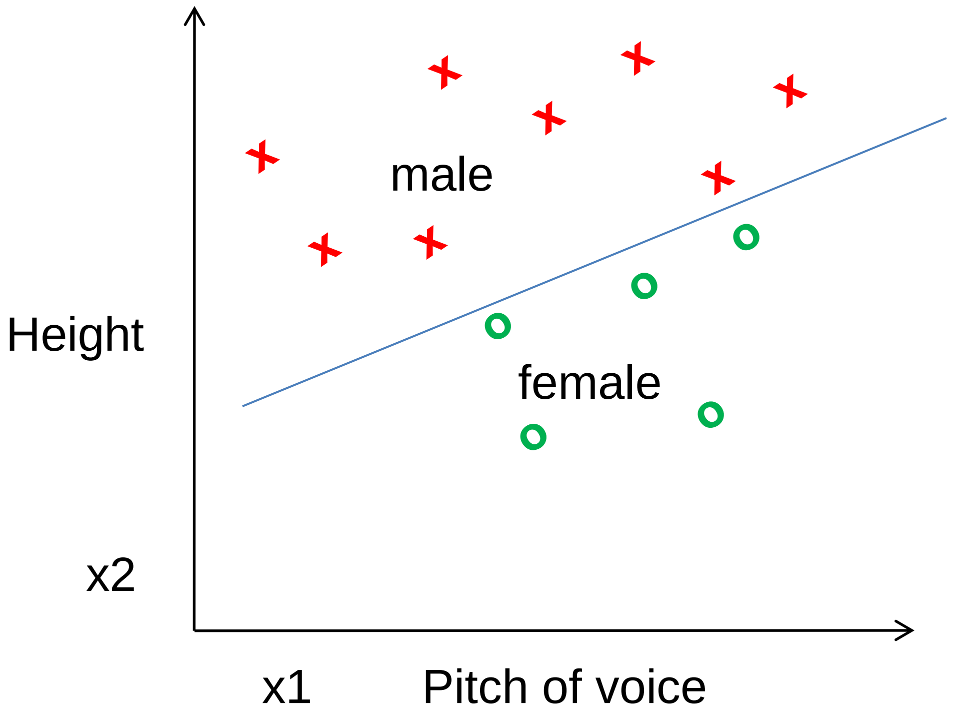
Using K-NN

- Simple, a good one to try first
- With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error



Classifiers: Logistic Regression

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



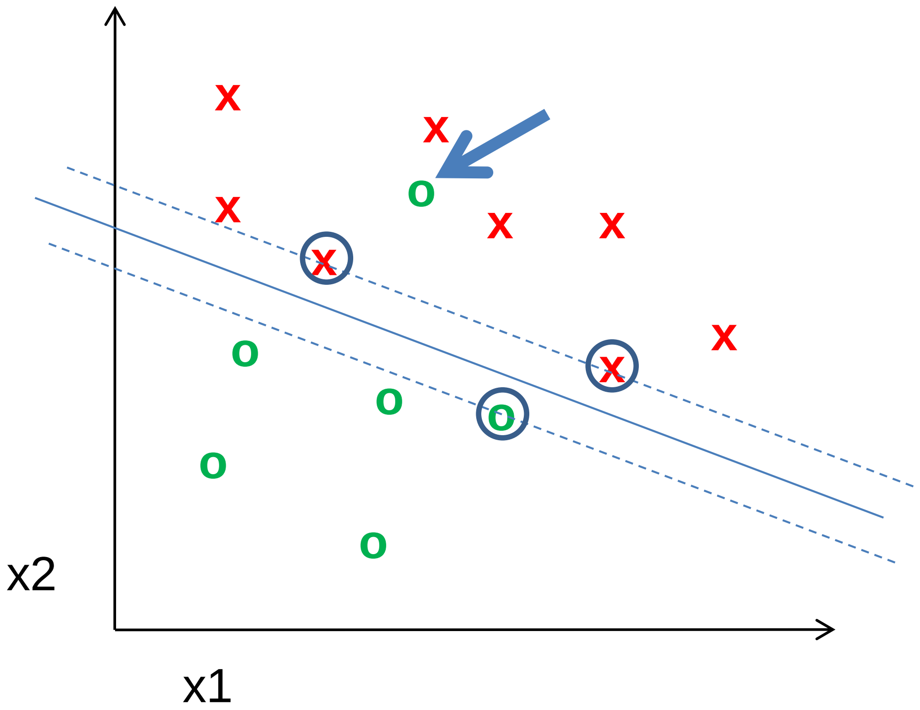
$$\log \frac{P(x_1, x_2 \mid y=1)}{P(x_1, x_2 \mid y=-1)} = \mathbf{w}^T \mathbf{x}$$

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

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

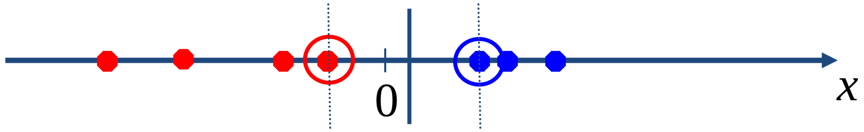


- Find a *linear function* to separate the classes:

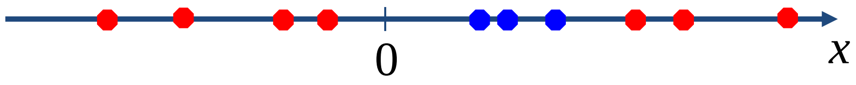
$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Nonlinear SVMs

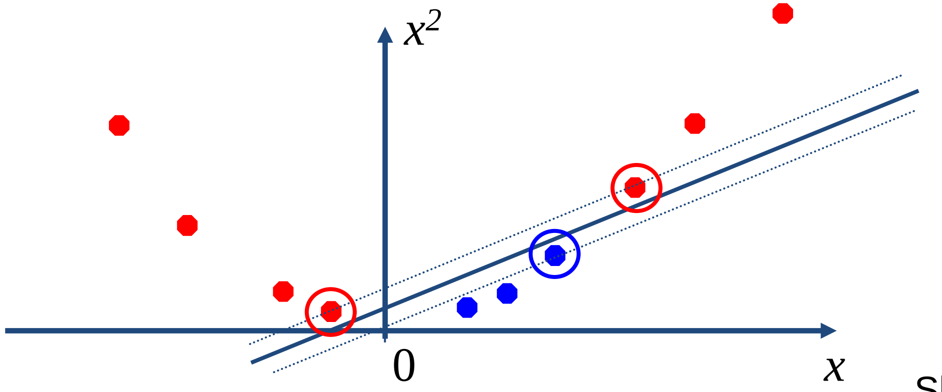
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



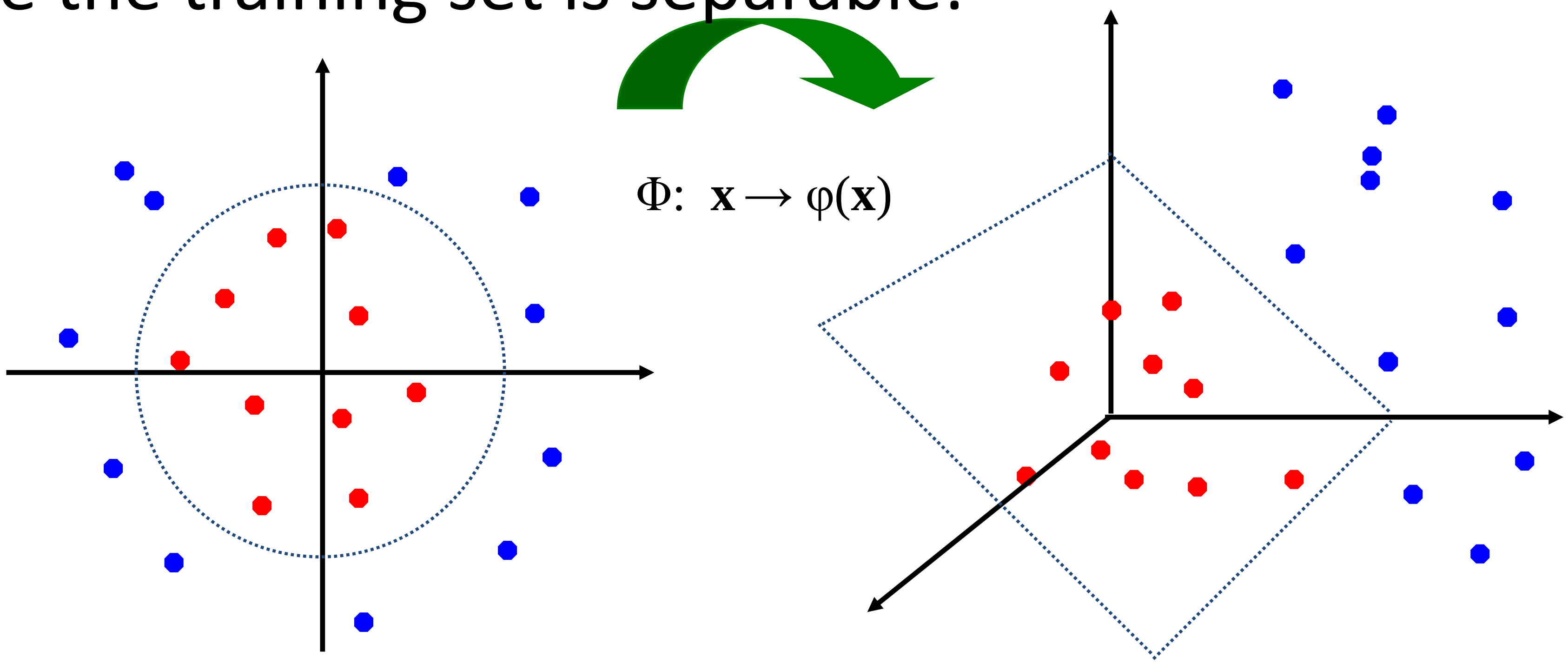
- We can map it to a higher-dimensional space:



Slide credit: Andrew Moore

Nonlinear SVMs

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



Slide credit: Andrew Moore

Summary: SVMs for image classification

1. Pick an image representation (in our case, bag of features)
2. Pick a kernel function for that representation
3. Compute the matrix of kernel values between every pair of training examples
4. Feed the kernel matrix into your favorite SVM solver to obtain support vectors and weights
5. At test time: compute kernel values for your test example and each support vector, and combine them with the learned weights to get the value of the decision function

Slide credit: L. Lazebnik

What about multi-class SVMs?

- Unfortunately, there is no “definitive” multi-class SVM formulation
- In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs
- One vs. others
  - Traning: learn an SVM for each class vs. the others
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- One vs. one
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM “votes” for a class to assign to the test example

Slide credit: L. Lazebnik

SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages: http://www.kernel-machines.org/software
  - Kernel-based framework is very powerful, flexible
  - SVMs work very well in practice, even with very small training sample sizes

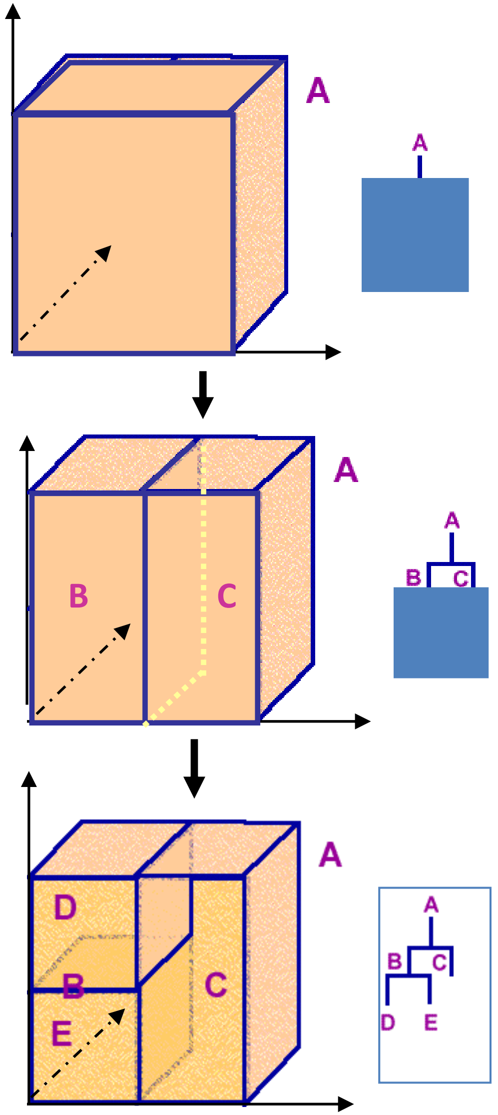
- Cons
  - No “direct” multi-class SVM, must combine two-class SVMs
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems

Decision Tree

- Classification Models: Many
- Why decision?
  - Relatively fast compared to other classification models
  - Obtain similar and sometimes better accuracy compared to other models
  - Simple and easy to understand
  - Can be converted into simple and easy to understand classification rules

Decision trees
-Binary decision trees

- Since each inequality that is used to split the input space is only based on one input variable.
- Each node draws a boundary that can be geometrically interpreted as a hyperplane perpendicular to the axis.

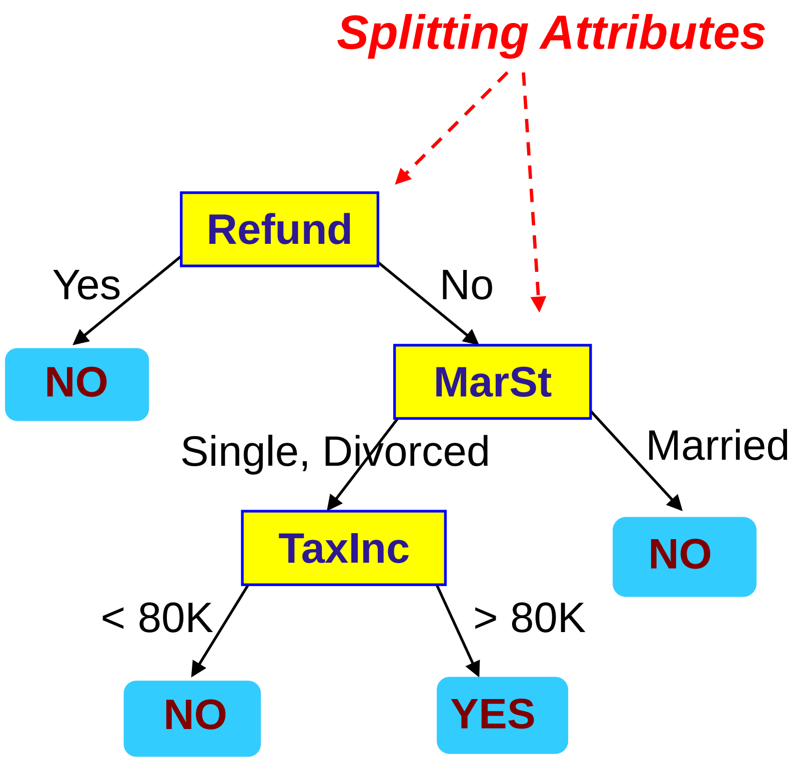


Model by Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

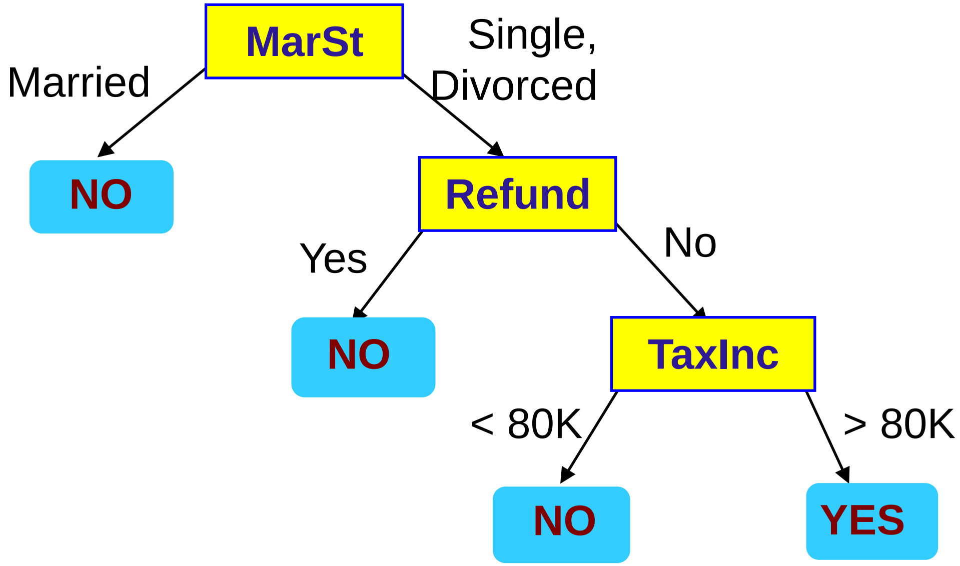


Model: Decision Tree

Another Example of Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

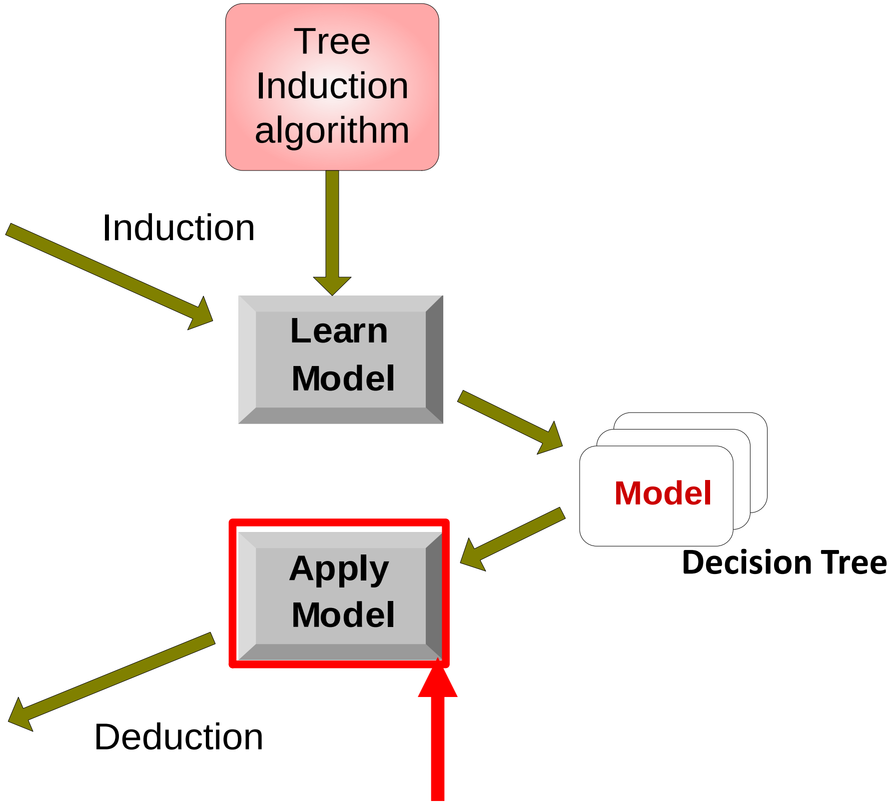


There could be more than one tree that fits the same data!

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

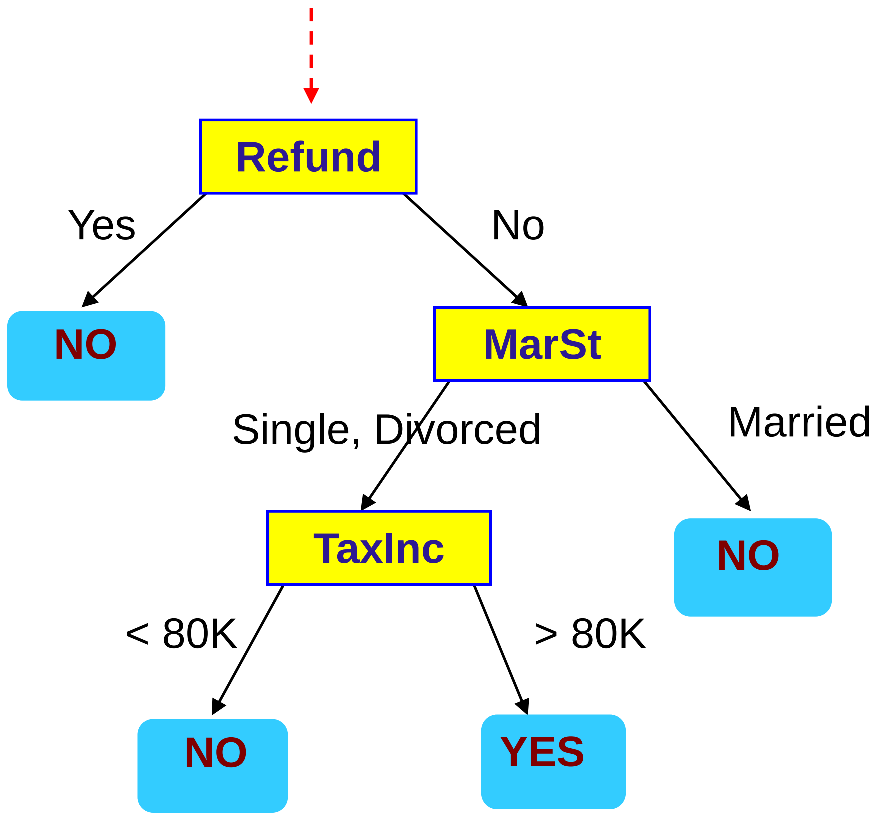


Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Apply Model to Test Data

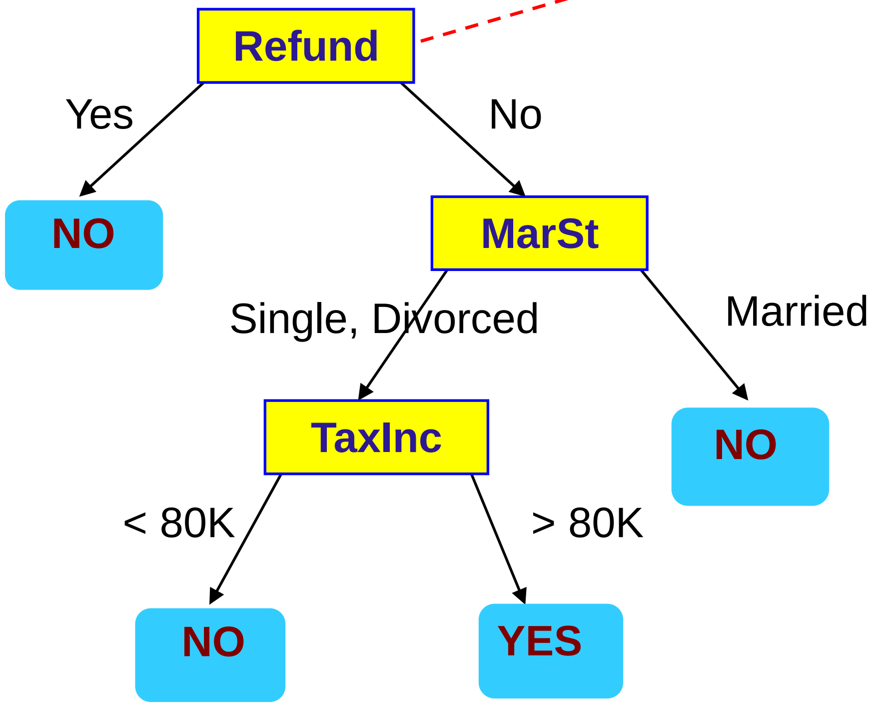
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



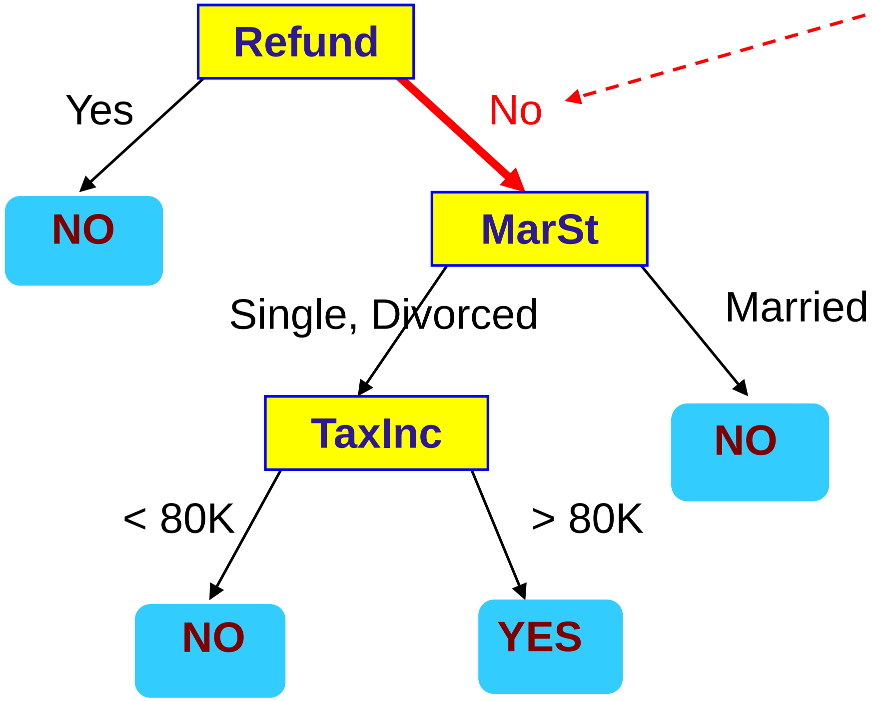
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

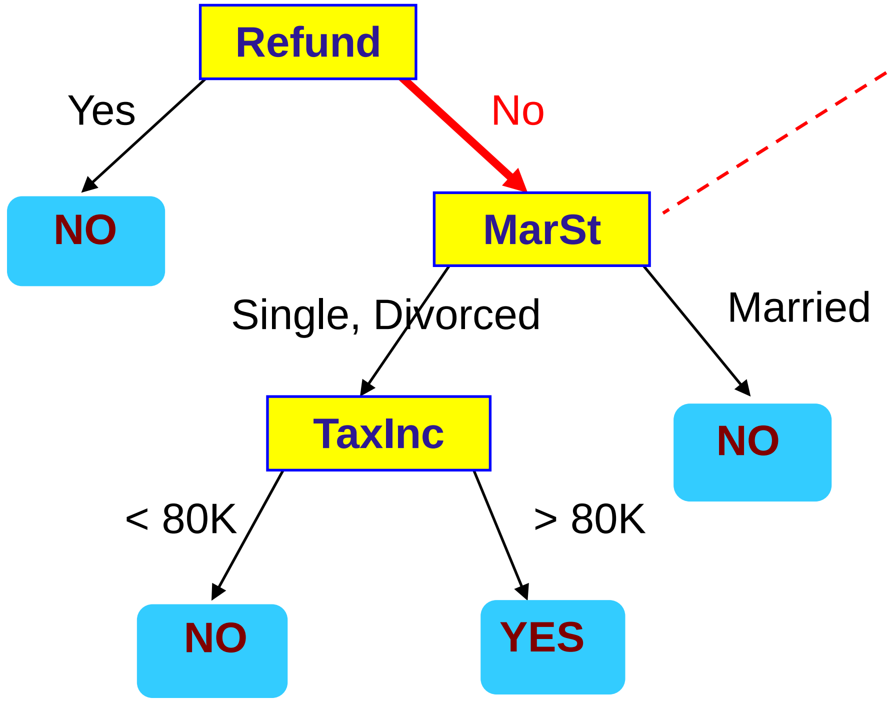
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



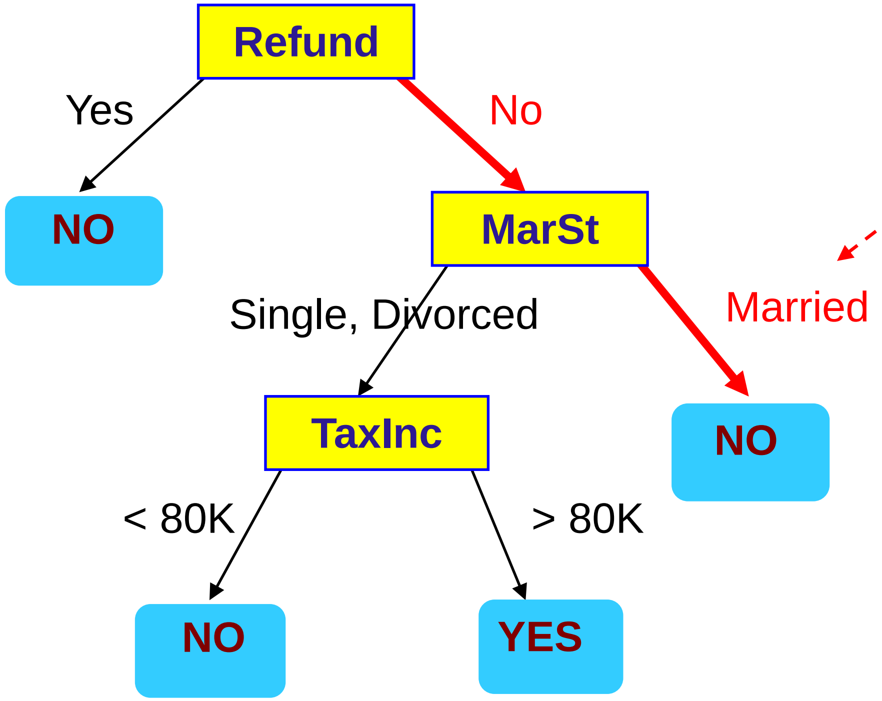
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



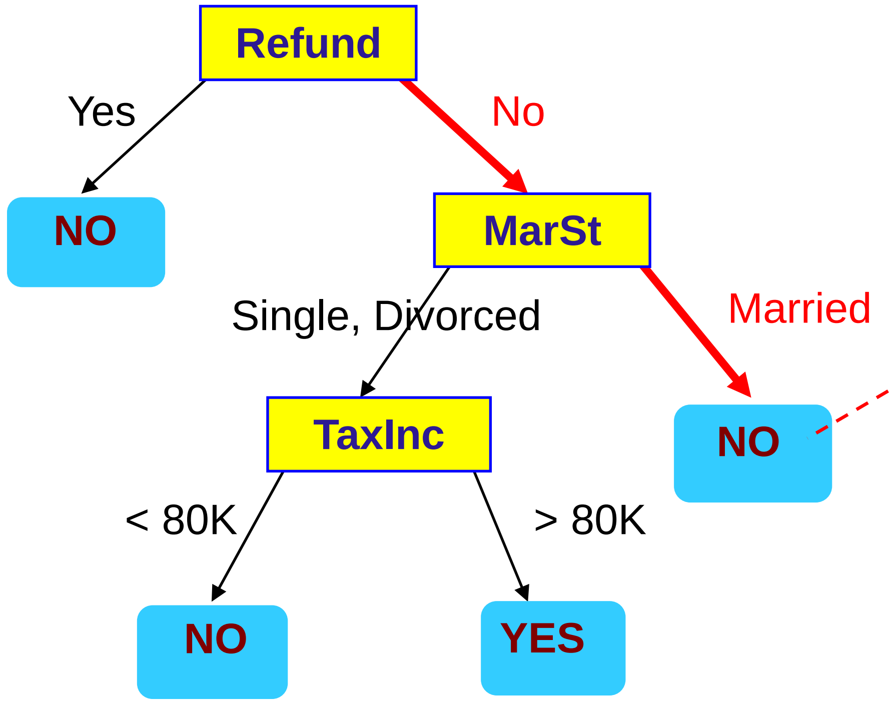
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



Test Data

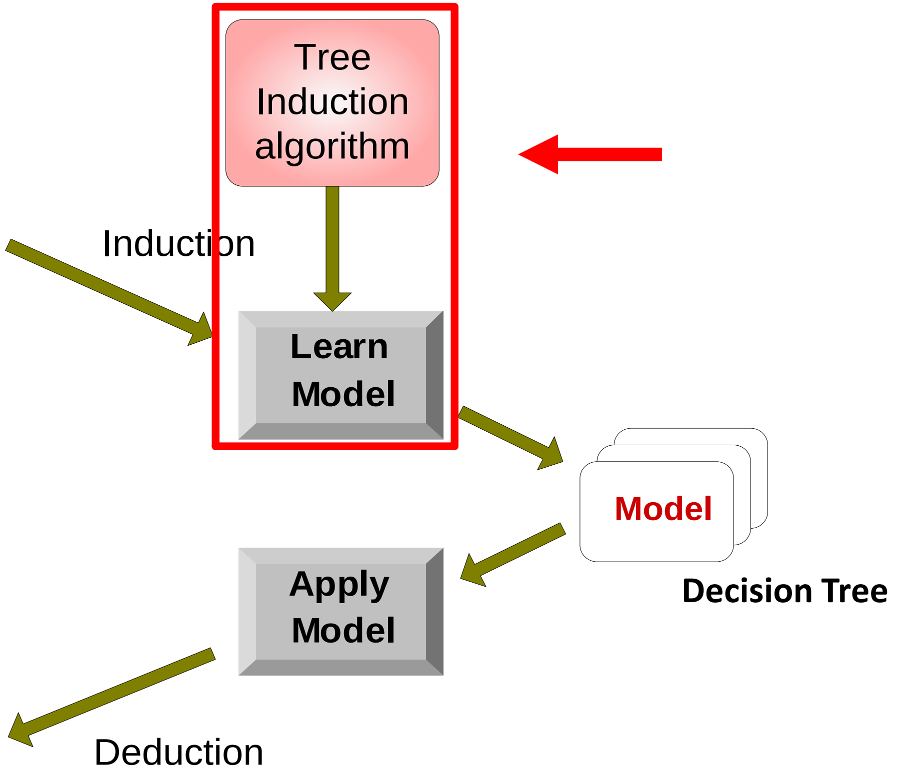
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Decision Tree Induction

- Many Algorithms:
  - Hunt’s Algorithm (one of the earliest)
  - CART
  - **ID3**, C4.5
  - SLIQ,SPRINT
- The basic idea behind any decision tree algorithm is as follows:
  - Choose the *best* attribute(s) to split the remaining instances and make that attribute a decision node
  - Repeat this process for recursively for each child
  - Stop when:
    - All the instances have the same target attribute value
    - There are no more attributes
    - There are no more instances

HMI
HUMAN MACHINE INTERACTION LABORATORY

Many classifiers to choose from

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

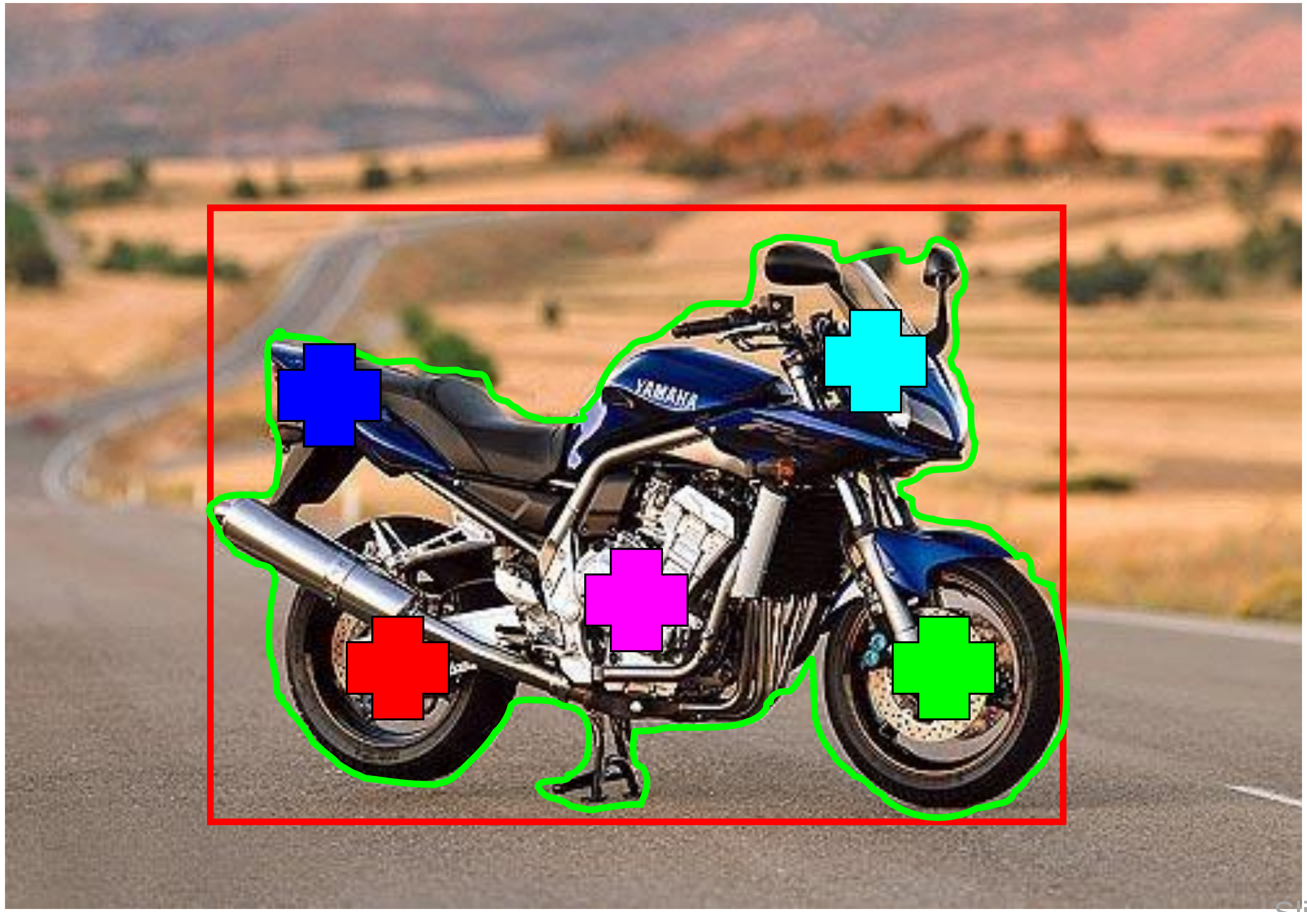
Which is the best one?

Slide credit: D. Hoiem

Recognition task and supervision

- Images in the training set must be annotated with the "correct answer" that the model is expected to produce

Contains a motorbike



Slide credit: L. Lazebnik

Spectrum of supervision

Less — More

Unsupervised

“Weakly” supervised

Fully supervised

Definition depends on task

Slide credit: L. Lazebnik

Generalization

Training set (labels known)

Test set (labels unknown)

- How well does a learned model generalize from the data it was trained on to a new test set?

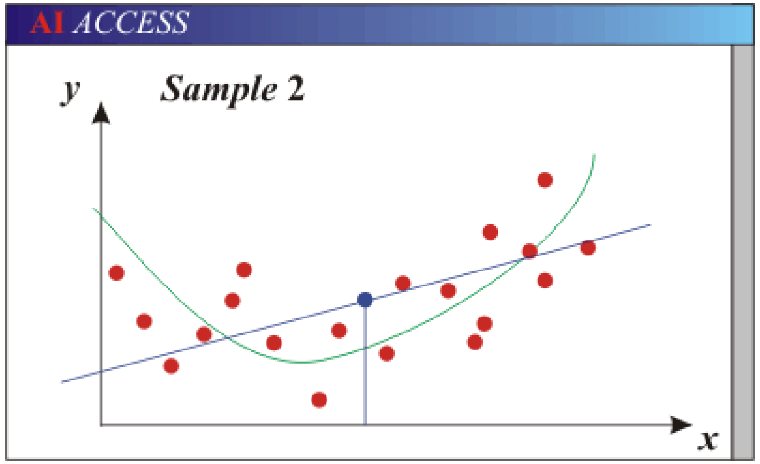
Slide credit: L. Lazebnik

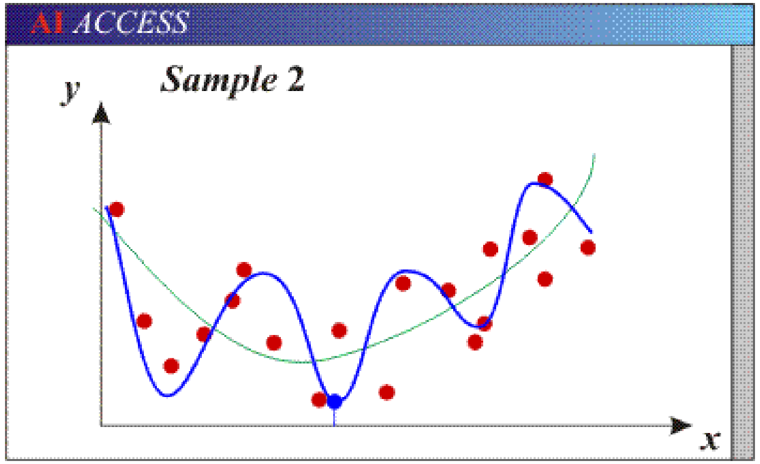
Generalization

- 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

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

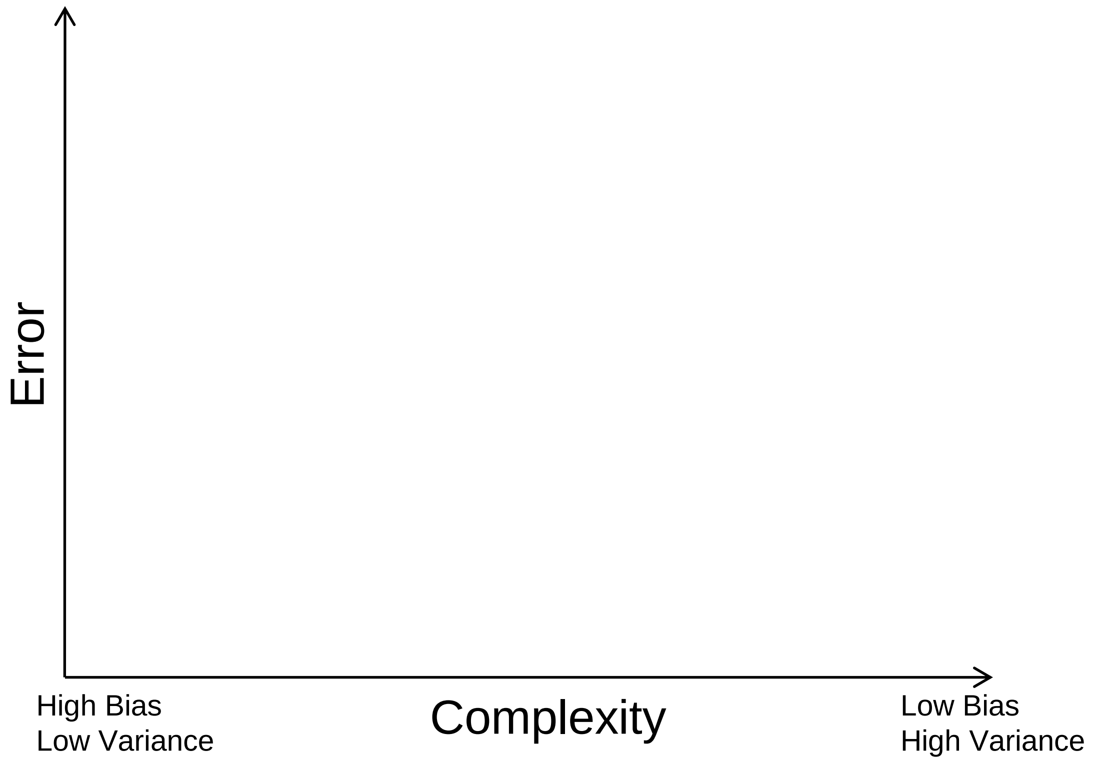
See the following for explanations of bias-variance (also Bishop's "Neural Networks" book):

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

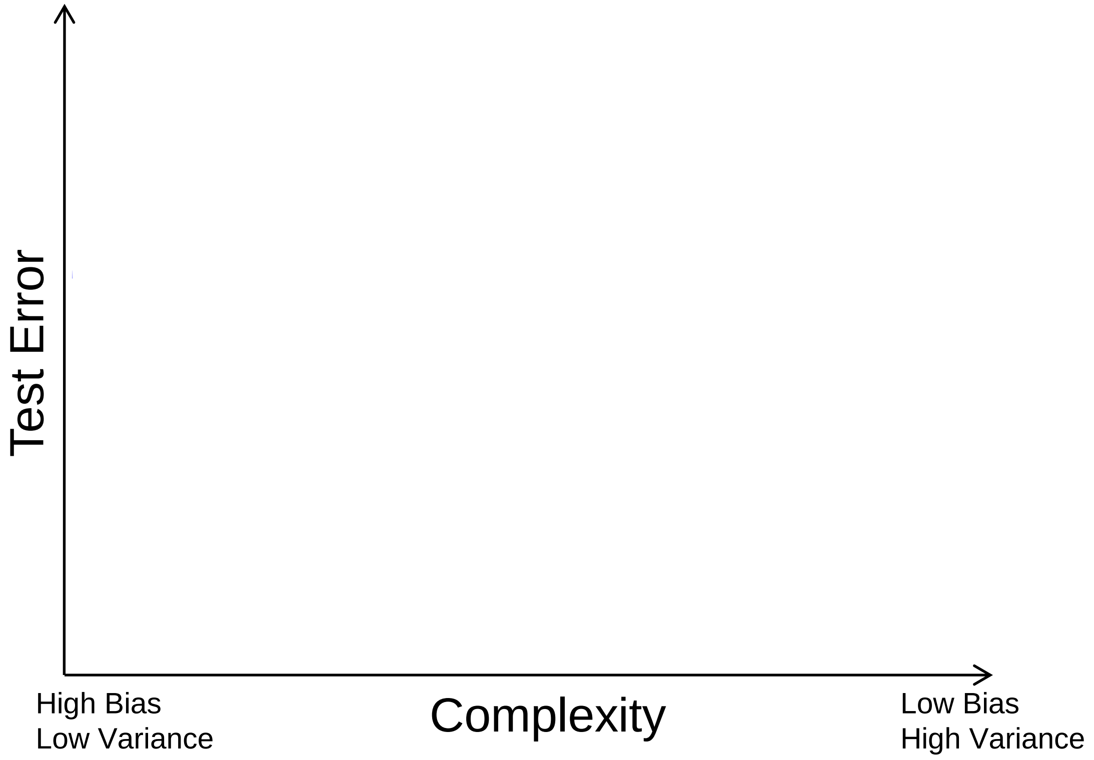
Slide credit: D. Hoiem

Bias-variance tradeoff

Underfitting **Overfitting**

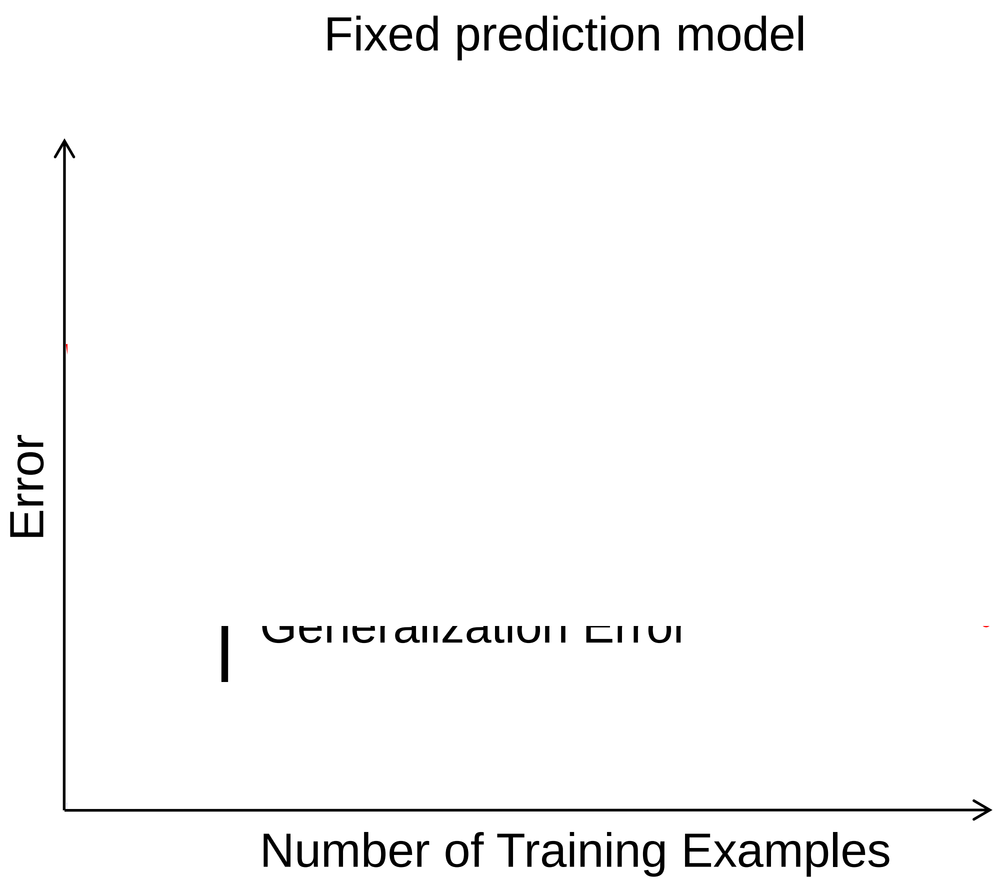


Bias-variance tradeoff



Slide credit: D. Hoiem

Effect of Training Size



Slide credit: D. Hoiem

The perfect classification algorithm

- Objective function: encodes the right loss for the problem
- Parameterization: makes assumptions that fit the problem
- Regularization: right level of regularization for amount of training data
- Training algorithm: can find parameters that maximize objective on training set
- Inference algorithm: can solve for objective function in evaluation

Slide credit: D. Hoiem

How to reduce variance?

- Choose a simpler classifier
- Regularize the parameters
- Get more training data

Slide credit: D. Hoiem

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

Ideals for a classification algorithm

- Objective function: encodes the right loss for the problem
- Parameterization: takes advantage of the structure of the problem
- Regularization: good priors on the parameters
- Training algorithm: can find parameters that maximize objective on training set
- Inference algorithm: can solve for labels that maximize objective function for a test example

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

Issues in Machine Learning

- What algorithms can approximate functions well and when?
- How does the number of training examples influence accuracy?
- How does the complexity of hypothesis representation impact it?
- How does noisy data influence accuracy?
- What are the theoretical limits of learnability?

Machine vs. Robot Learning

Machine Learning

- Learning in vaccum
- Statistically well-behaved data
- Mostly off-line
- Informative feed-back
- Computational time not an issue
- Hardware does not matter
- Convergence proof

Robot Learning

- Embedded learning
- Data distribution not homegeneous
- Mostly on-line
- Qualitative and sparse feed-back
- Time is crucial
- Hardware is a priority
- Empirical proof