Lecture 06: Machine Learning

Instructor: Dr. Hossam Zawbaa

Machine Learning Problems

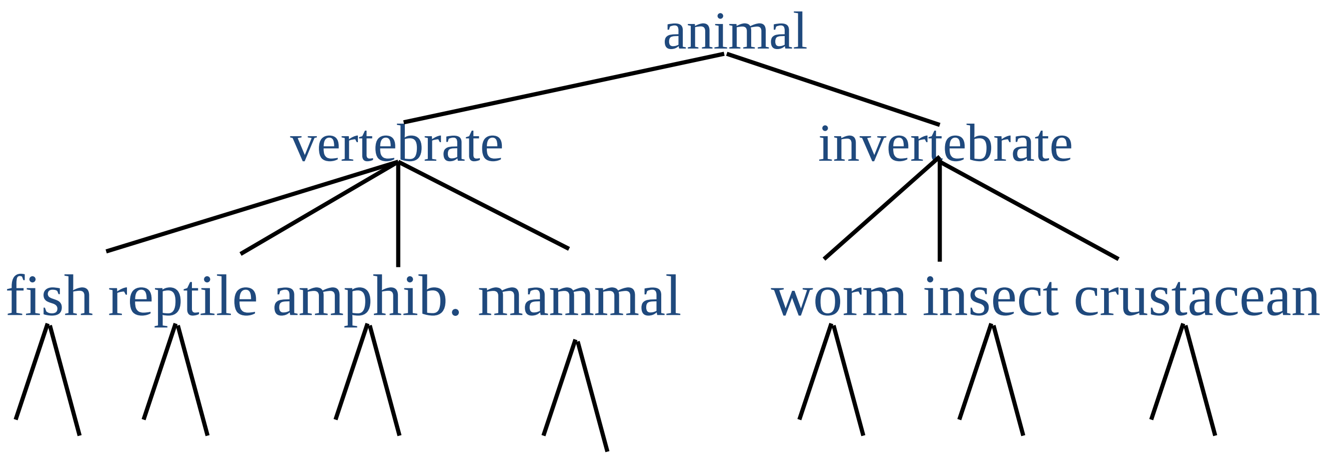
	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Clustering Strategies

- **K-means**
  - Iteratively re-assign points to the nearest cluster center.
- **Hierarchical Agglomerative clustering**
  - Start with each point as its own cluster and iteratively merge the closest clusters.
- **Mean-shift clustering**
  - Estimate modes of probability density function (pdf).
- **Spectral clustering**
  - Split the nodes in a graph based on assigned links with similarity weights.

Hierarchical Clustering

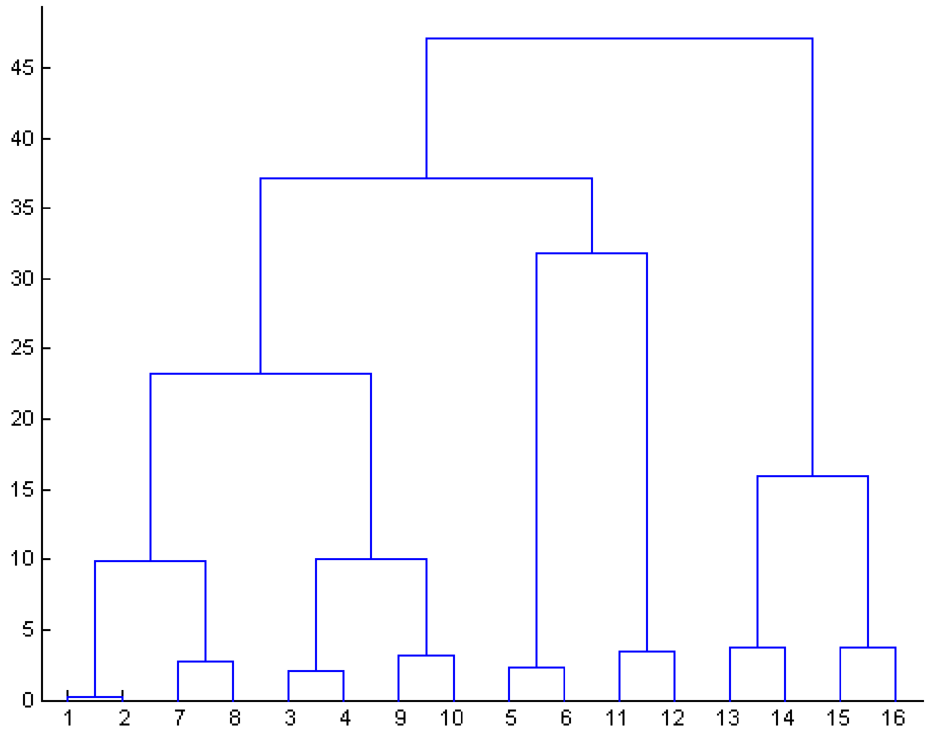
- Build a tree-based hierarchical taxonomy from a set of images.



Hierarchical Clustering algorithms

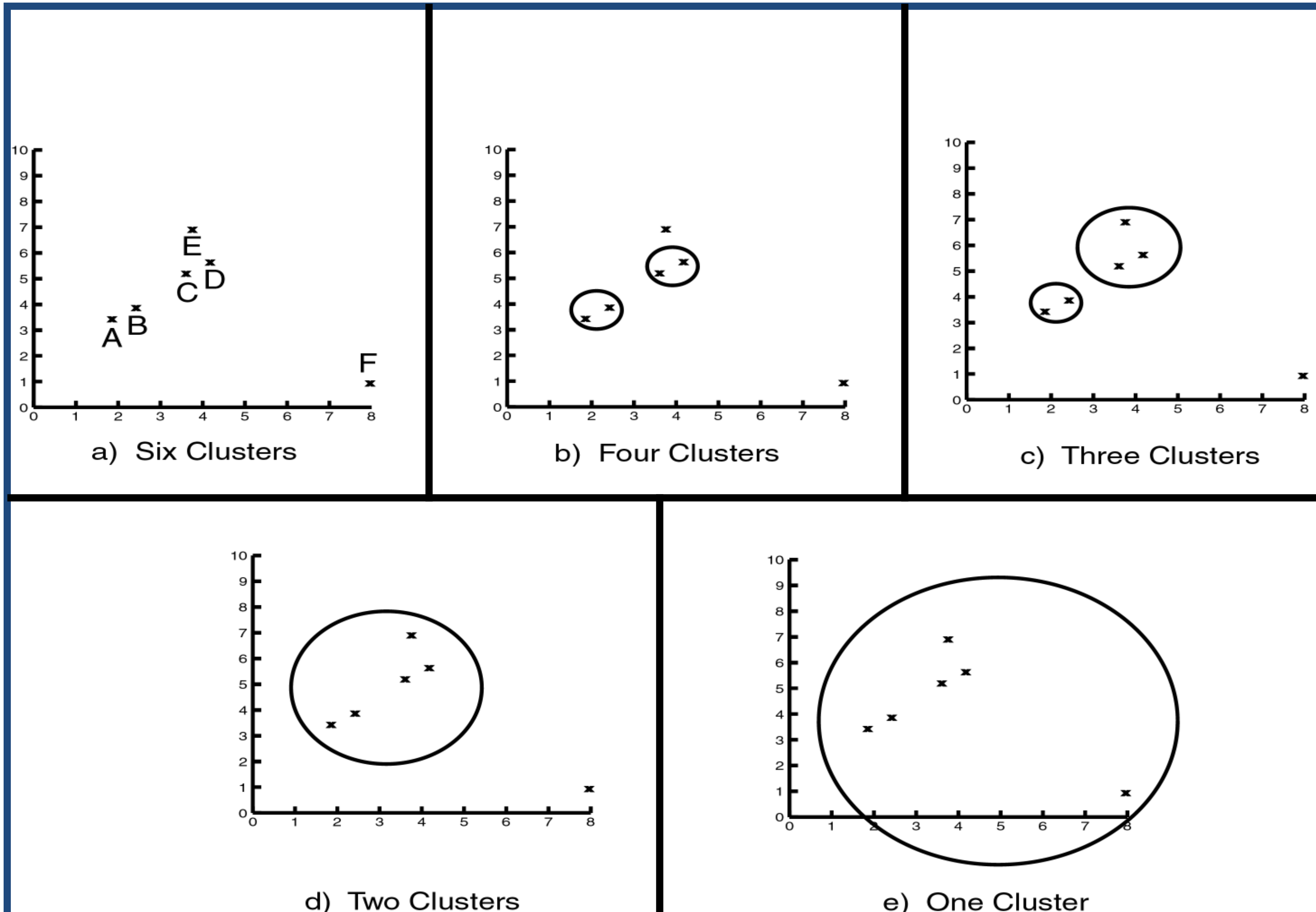
- **Agglomerative (bottom-up):**
  - Start with each image being a single cluster.
  - At the end, all images belong to the same cluster.
- **Divisive (top-down):**
  - Start with all images belong to the same cluster.
  - At the end, each node forms a cluster on its own.
- Does not require the number of clusters ***k*** in advance.
- Needs a termination condition.

Hierarchical Clustering



- This produces a binary tree or *dendrogram*
- The final cluster is the root and each data item is a leaf
- The height of the bars indicate how close the items are

Levels of Clustering



Machine Learning Problems

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

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

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

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

ML Steps

Training

Training Images

Image Features → Training → Learned model

Training Labels → Training

Testing

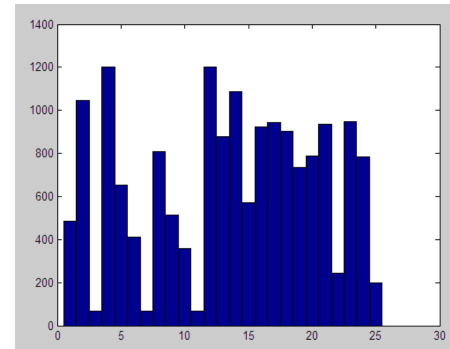
Test Image

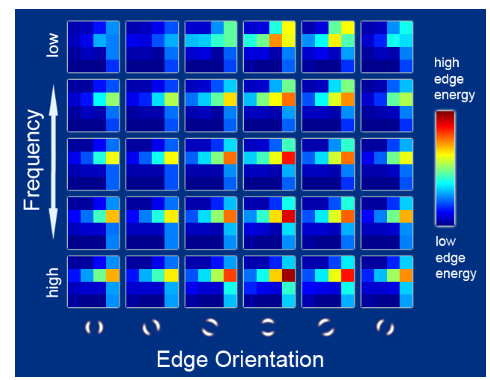
Image Features → Learned model → Prediction

Slide credit: D. Hoiem and L. Lazebnik

Features

- Raw pixels
- Histograms
- SIFT descriptors
- ...





Slide credit: L. Lazebnik

Many classifiers to choose from

- Support Vector Machine
- Artificial Neural networks
- K-nearest neighbor
- Decision Trees
- Naïve Bayes
- Bayesian network
- Logistic regression
- Random Forests
- Etc.

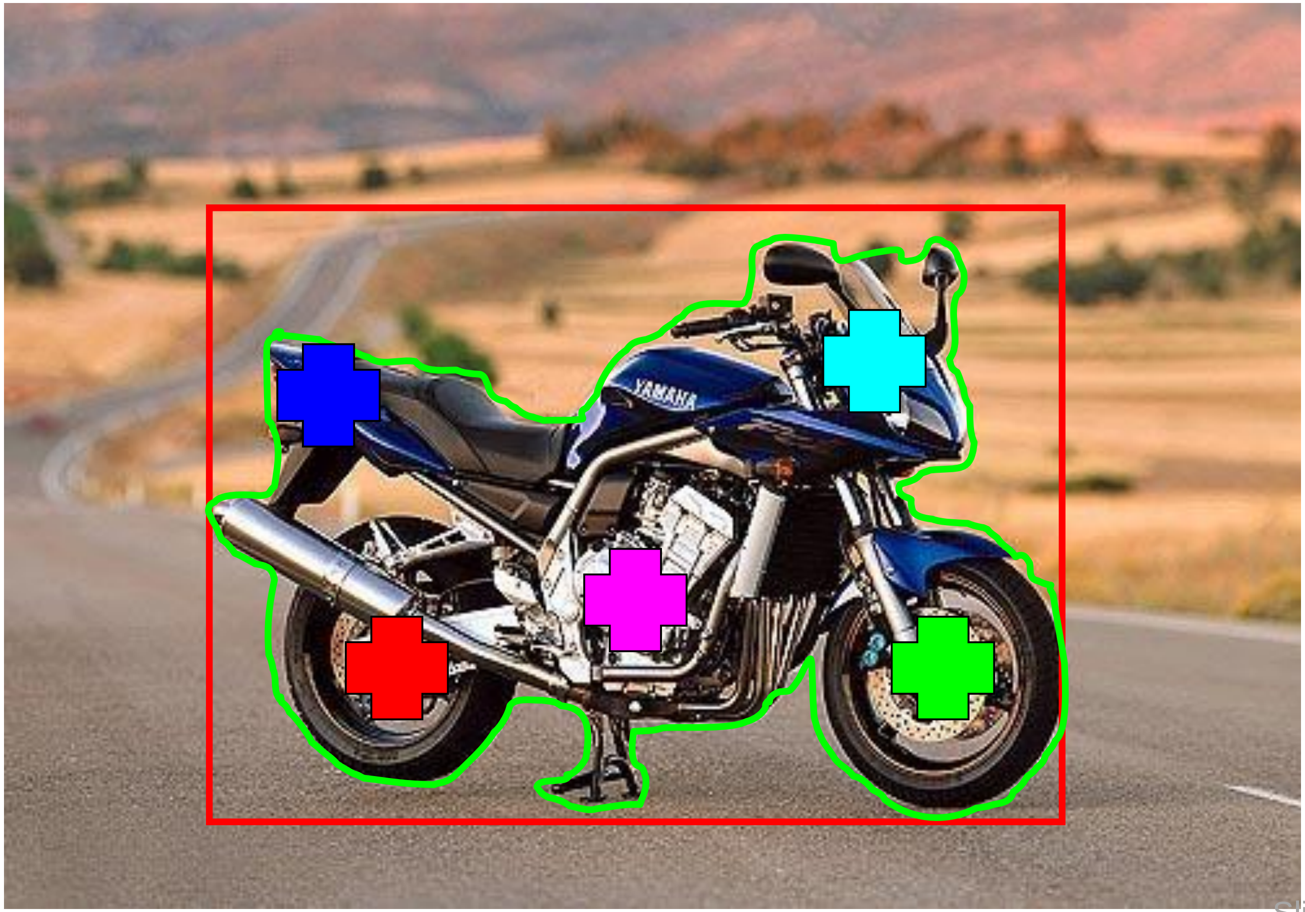
Which is the best one?

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)

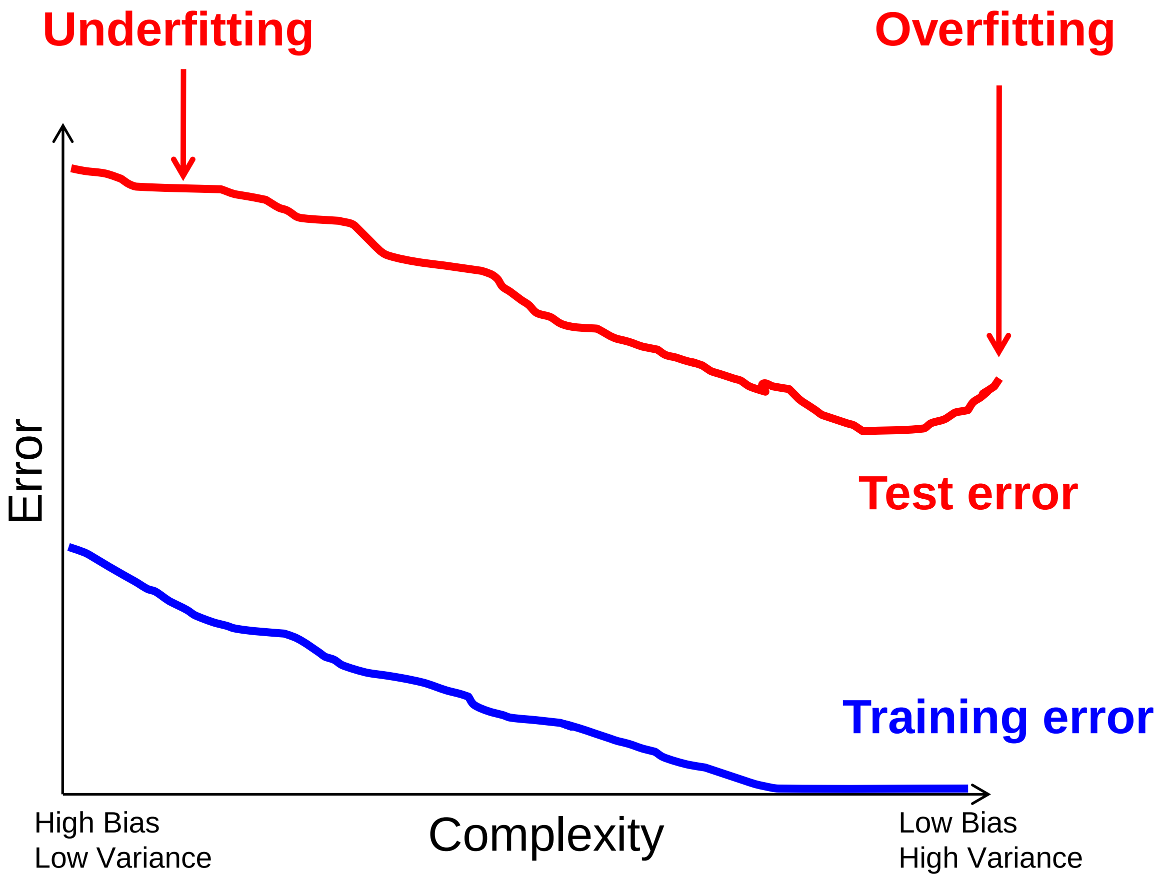
- How well does a learned model generalize from the data it was trained on to a new test set?

Slide credit: L. Lazebnik

Generalization

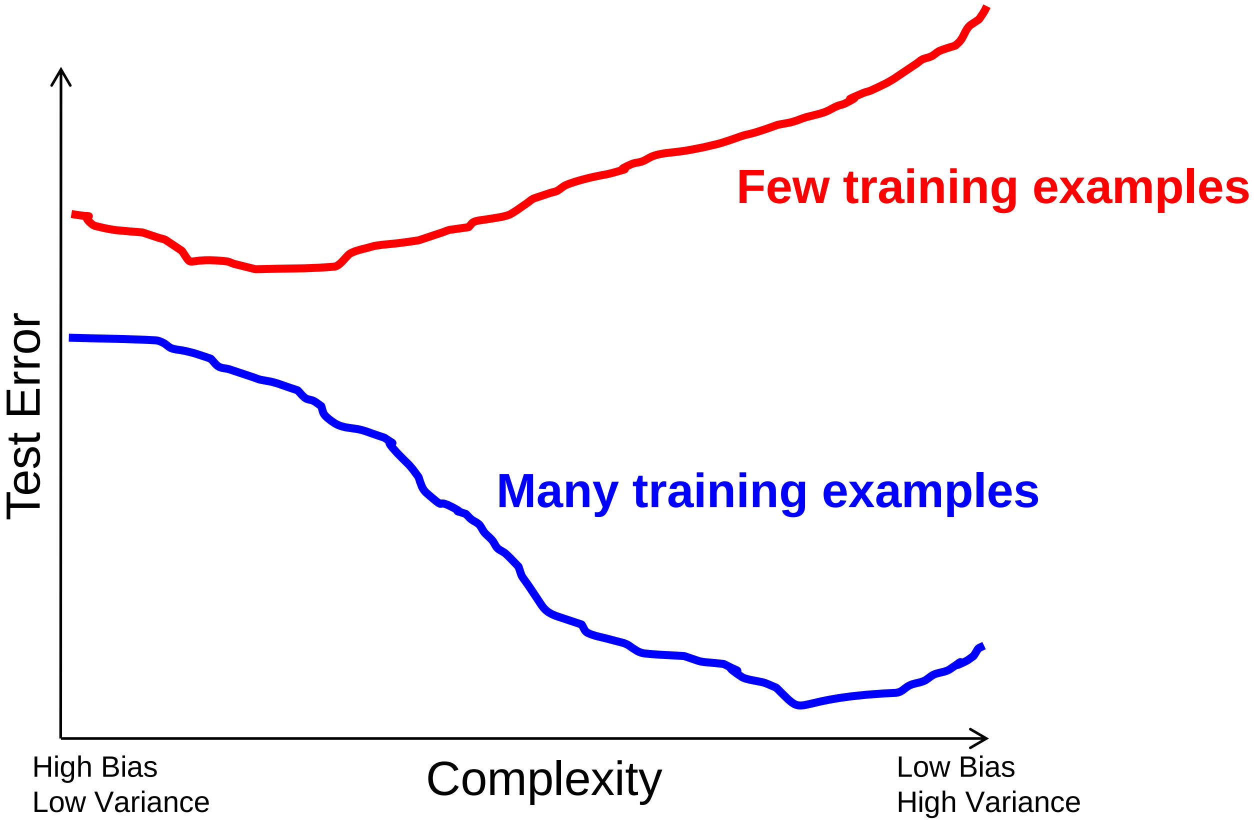
- Components of generalization error
  - **Bias:** how much the average model (**prediction**) over all training sets differ from the true model (**actual**)?
    - **Error due to inaccurate assumptions/simplifications made by the model**
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model is too "simple" to represent all the relevant class characteristics
  - High training error and high test error
- **Overfitting:** model is too "complex" and fits irrelevant characteristics (noise) in the data
  - Low training error and high test error

Bias-Variance Trade-off



Slide credit: D. Hoiem

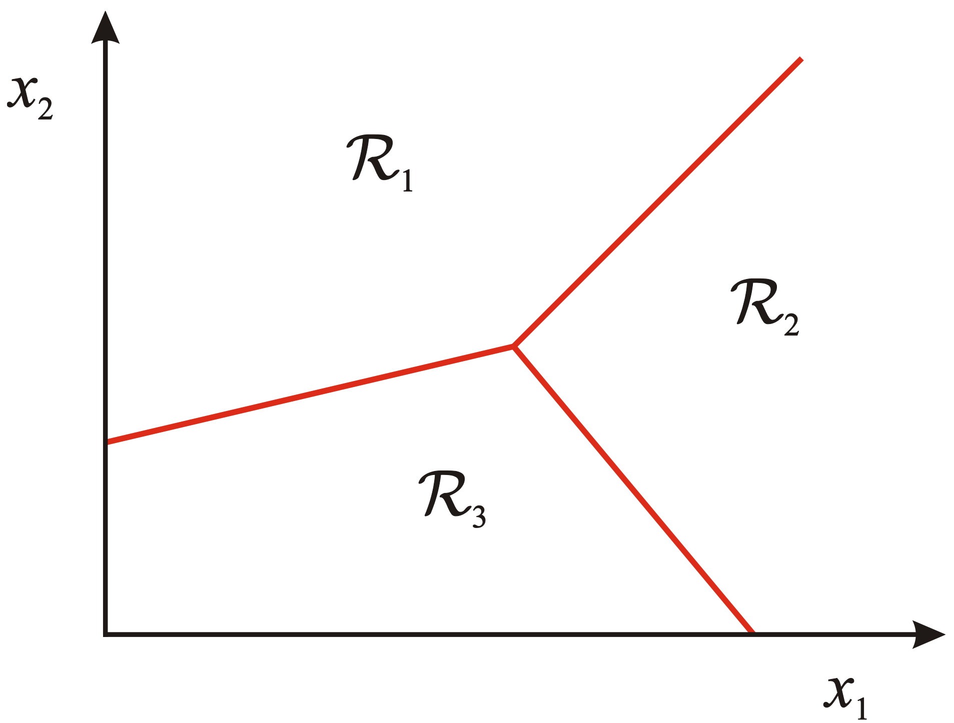
Bias-Variance Trade-off



Slide credit: D. Hoiem

Classification

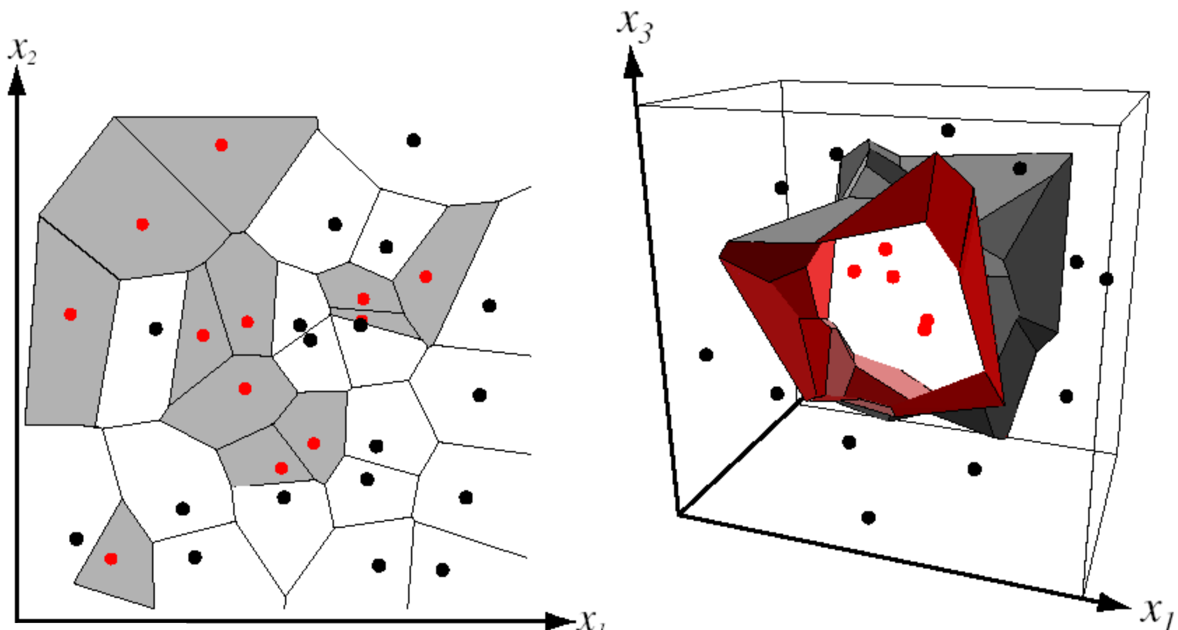
- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



Slide credit: L. Lazebnik

Nearest Neighbor Classifier

- Assign label of nearest training data point to each test data point

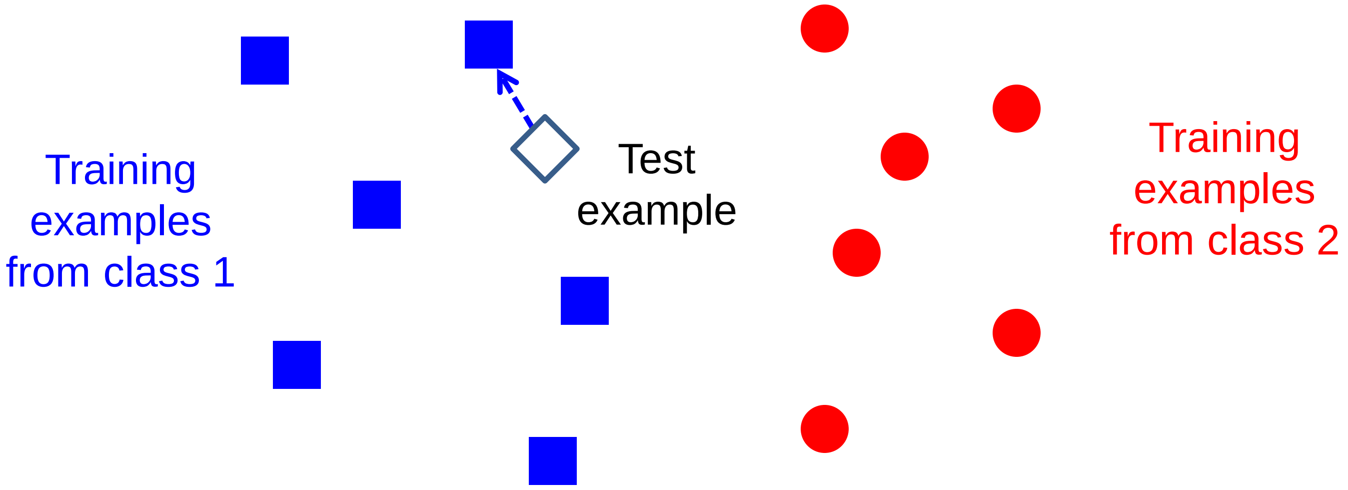


from Duda *et al.*

Voronoi partitioning of feature space
for two-category 2D and 3D data

Source: D. Lowe

Classifiers: Nearest neighbor

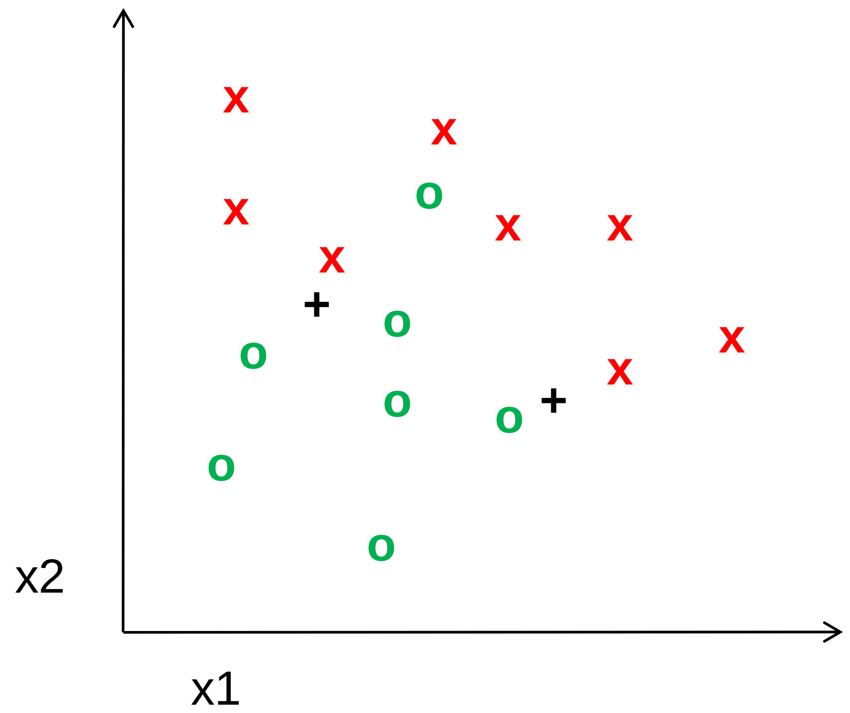


f(**x**) = label of the training example nearest to **x**

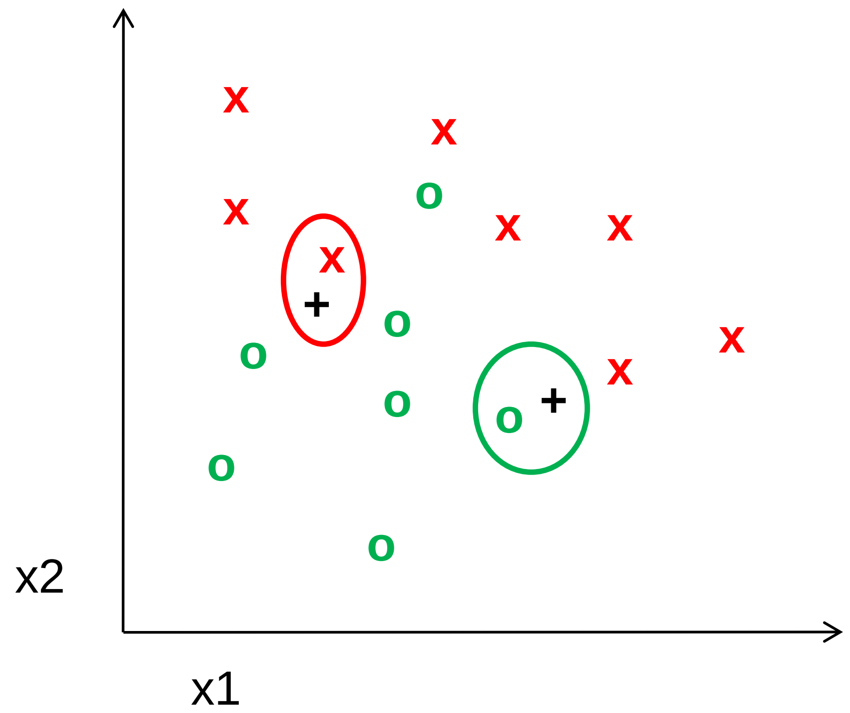
- **All we need is a distance function for our inputs**
- **No training required!**

Slide credit: L. Lazebnik

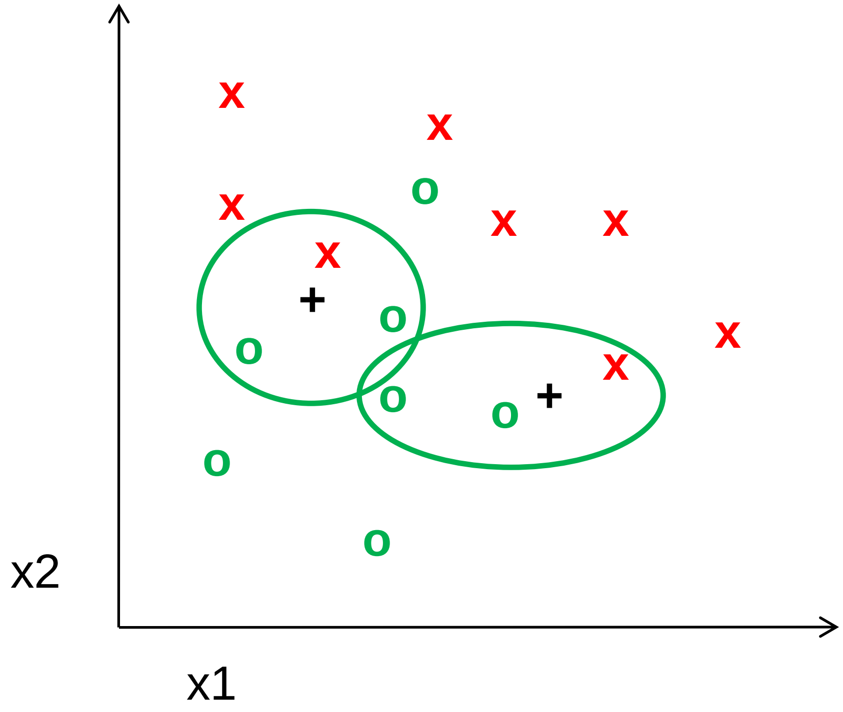
K-nearest neighbor



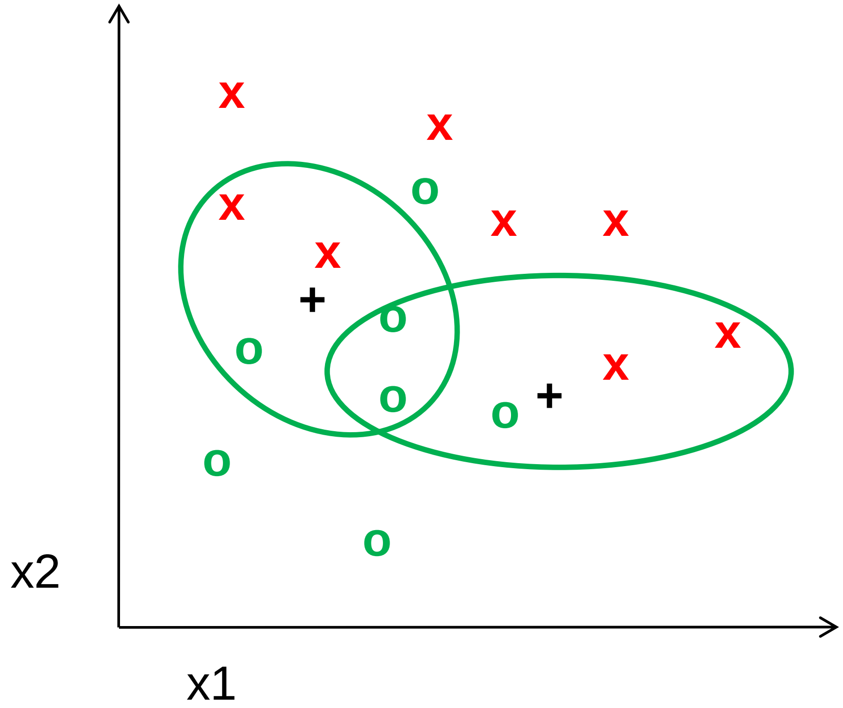
1-nearest neighbor



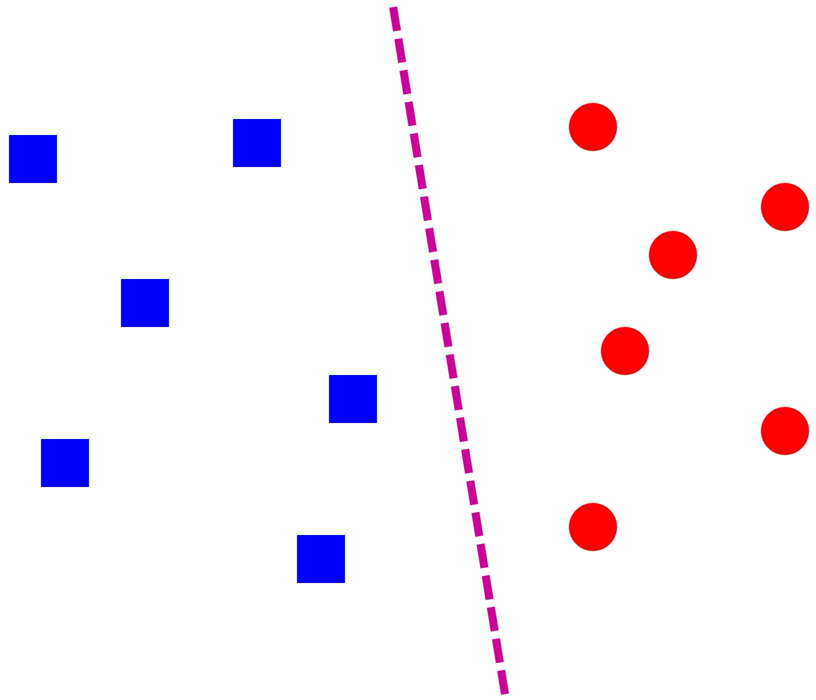
3-nearest neighbor



5-nearest neighbor



Classifiers: Linear

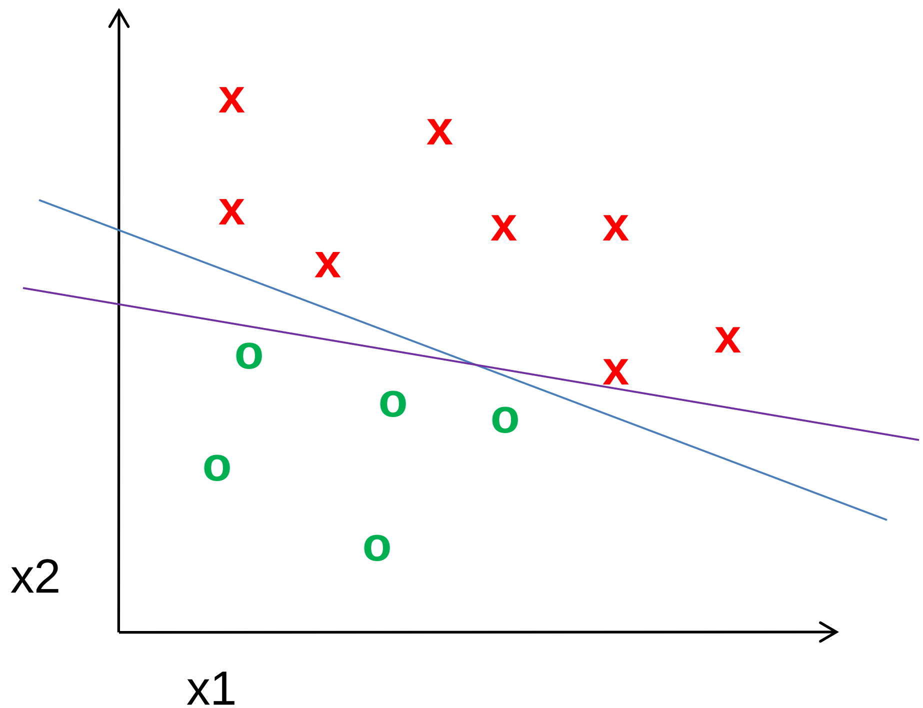


- Find a *linear function* to separate the classes:

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

Slide credit: L. Lazebnik

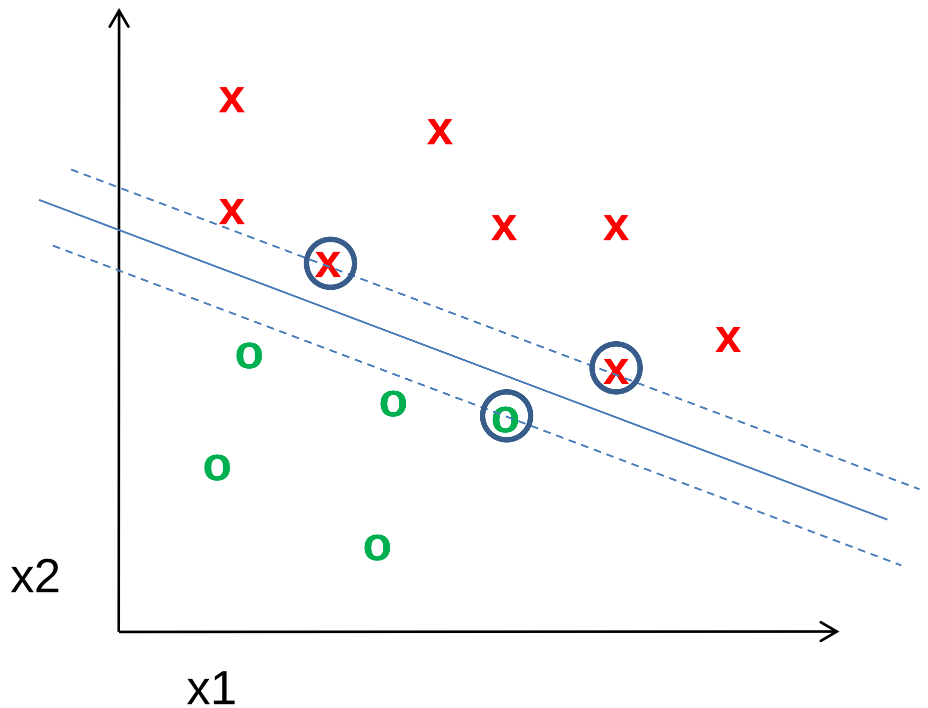
Classifiers: Linear ANN



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

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

Classifiers: Linear SVM

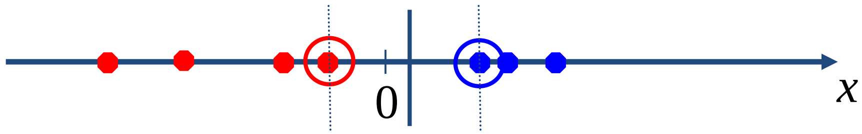


- Find a *linear function* to separate the classes:

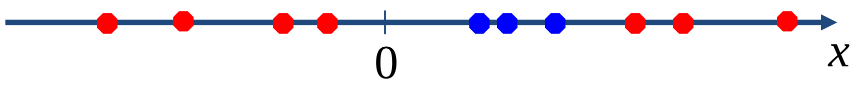
$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Nonlinear SVMs

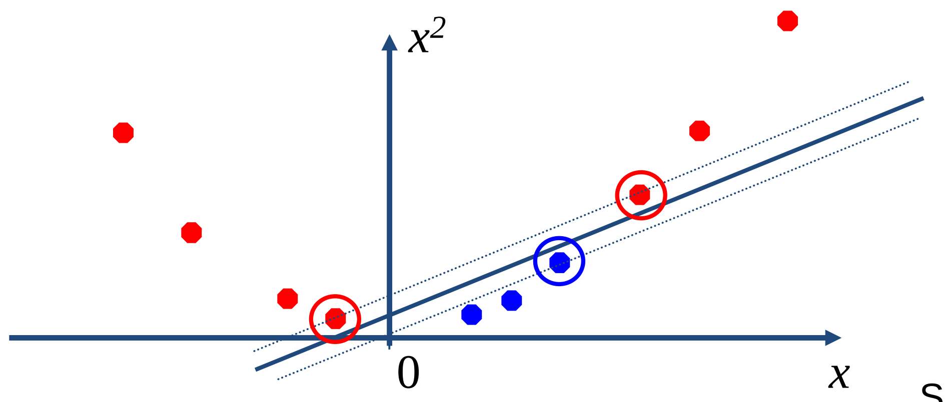
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



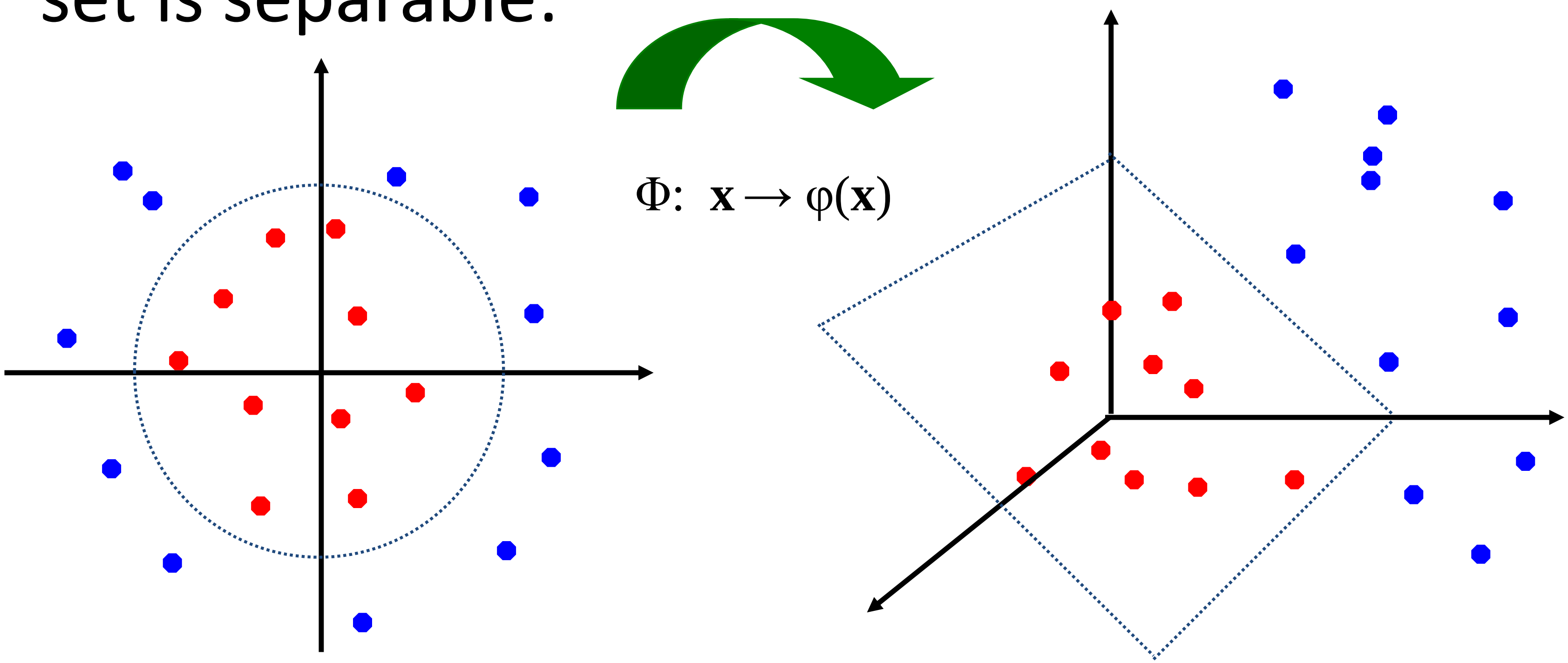
- We can map it to a higher-dimensional space:



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

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
  - Training: 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