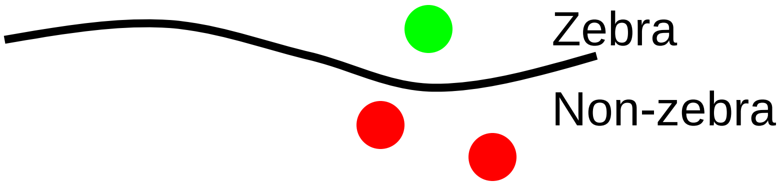
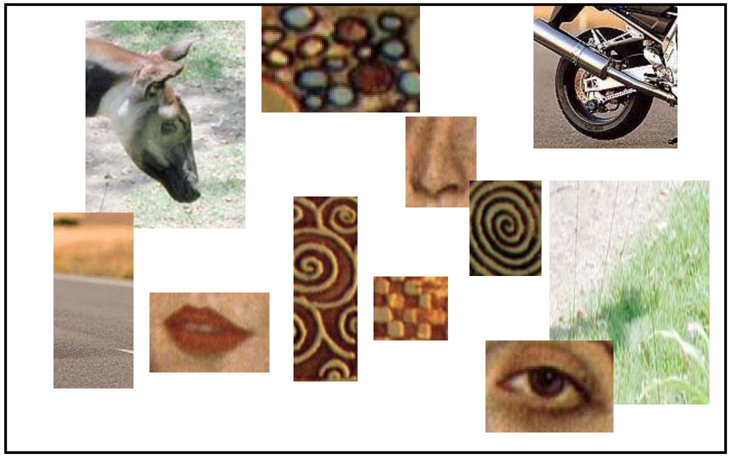
Discriminative and generative methods for bags of features

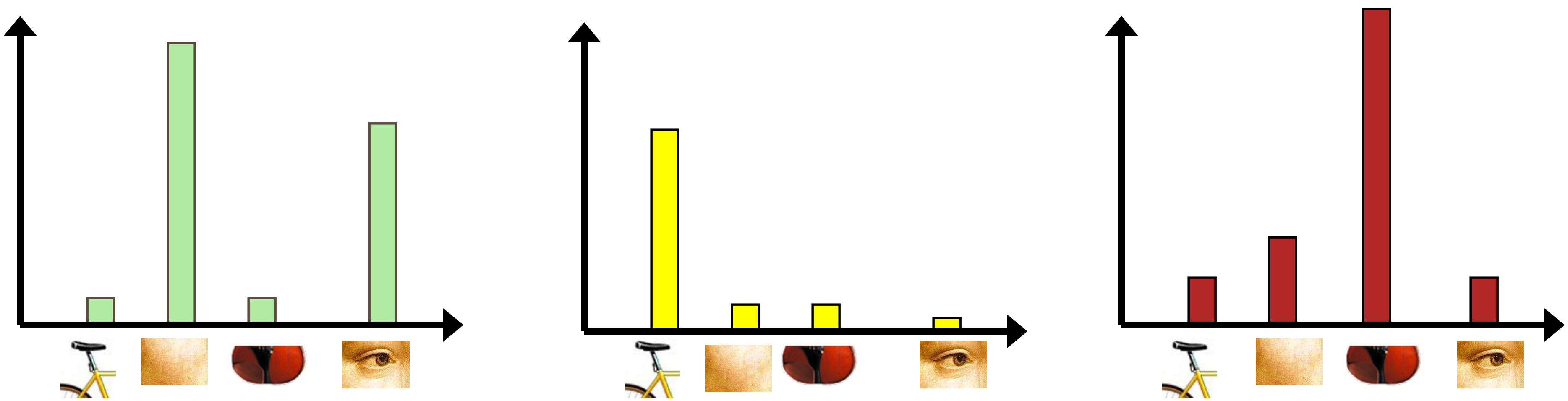




Many slides adapted from Fei-Fei Li, Rob Fergus, and Antonio Torralba

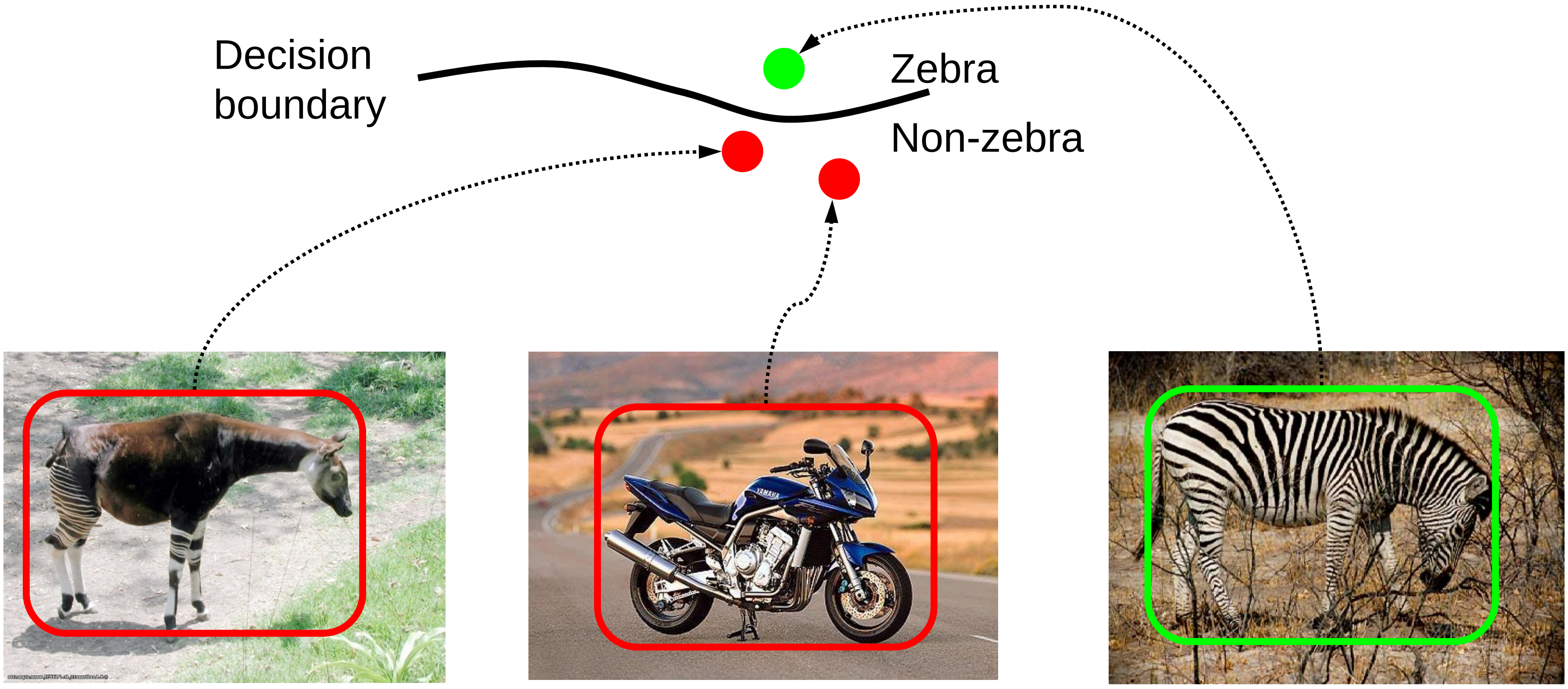
Image classification

- Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?



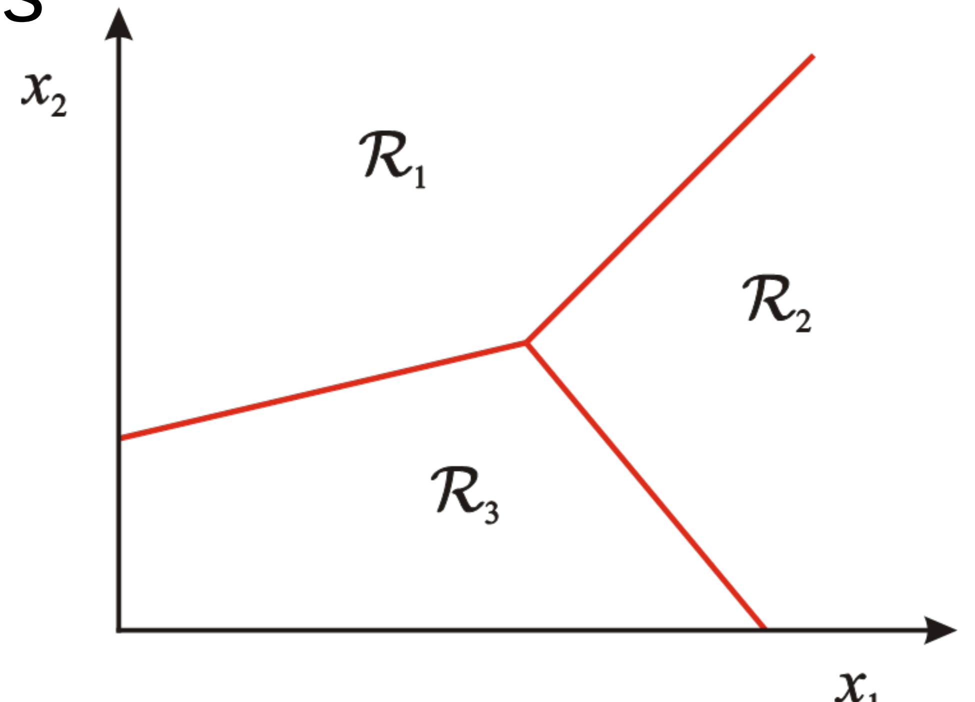
Discriminative methods

- Learn a decision rule (classifier) assigning bag-of-features representations of images to different classes



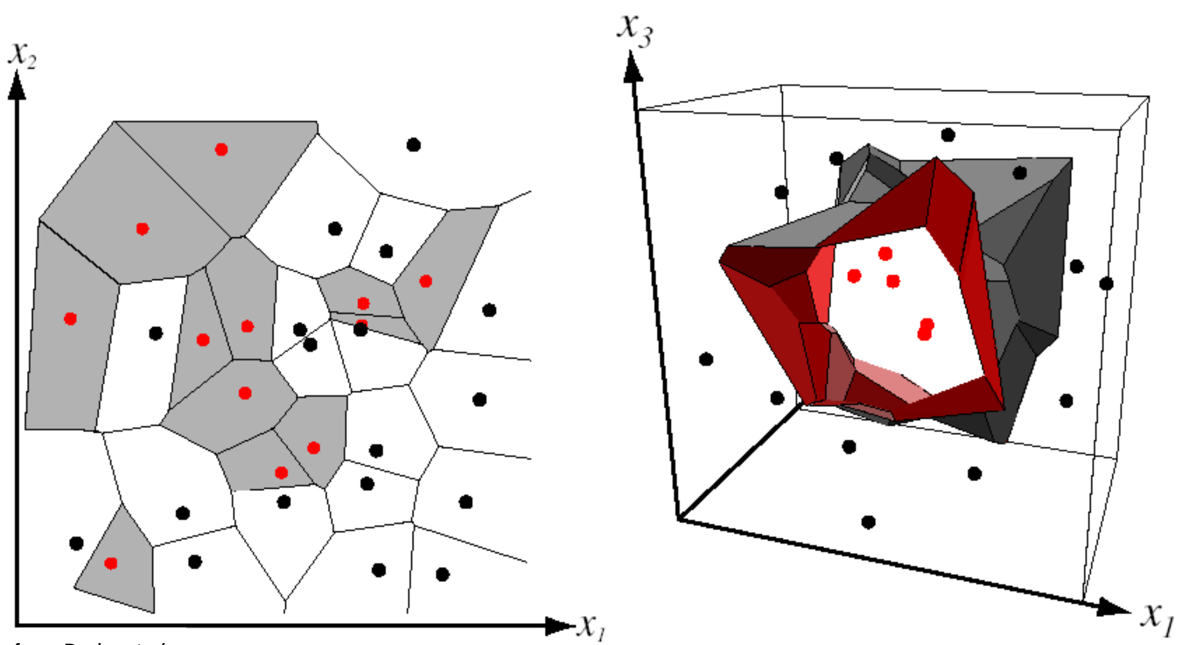
Classification

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



Nearest Neighbor Classifier

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



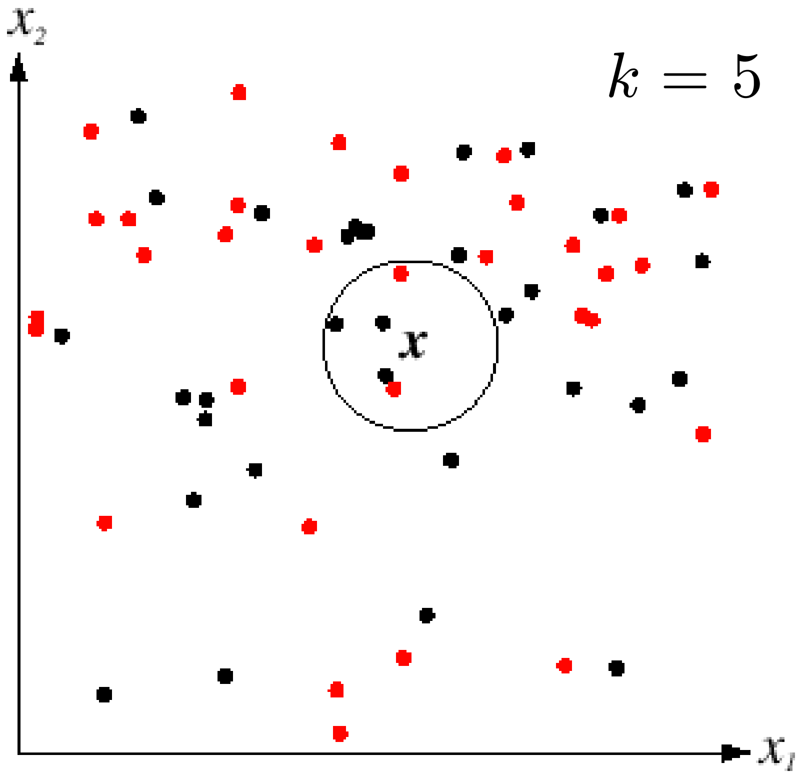
from Duda *et al.*

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

Source: D. Lowe

K-Nearest Neighbors

- For a new point, find the k closest points from training data
- Labels of the k points "vote" to classify
- Works well provided there is lots of data and the distance function is good



Source: D. Lowe

Functions for comparing histograms

- L1 distance

$$D(h_1, h_2) = \sum_{i=1}^{N} | h_1(i) - h_2(i) |$$

- χ^2 distance

$$D(h_1, h_2) = \sum_{i=1}^{N} \frac{\left(h_1(i) - h_2(i)\right)^2}{h_1(i) + h_2(i)}$$

- Quadratic distance (*cross-bin*)

$$D(h_1, h_2) = \sum_{i,j} A_{ij} (h_1(i) - h_2(j))^2$$

Jan Puzicha, Yossi Rubner, Carlo Tomasi, Joachim M. Buhmann: Empirical Evaluation of Dissimilarity Measures for Color and Texture. ICCV 1999

Earth Mover's Distance

- Each image is represented by a *signature S* consisting of a set of *centers* $\{m_i\}$ and *weights* $\{w_i\}$
- Centers can be codewords from universal vocabulary, clusters of features in the image, or individual features (in which case quantization is not required)
- Earth Mover's Distance has the form

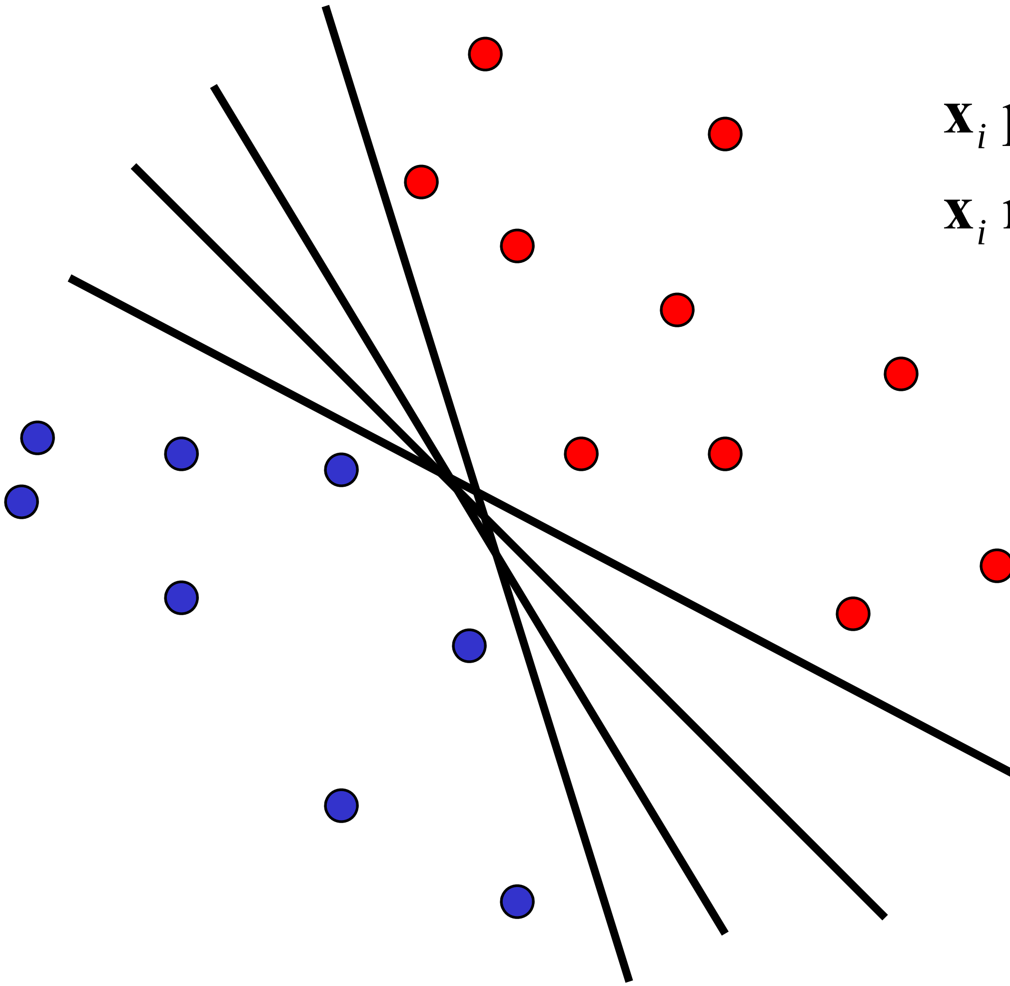
$$EMD(S_1, S_2) = \sum_{i,j} \frac{f_{ij}\, d(m_{1i}, m_{2j})}{f_{ij}}$$

where the *flows* f_{ij} are given by the solution of a *transportation problem*

Y. Rubner, C. Tomasi, and L. Guibas: A Metric for Distributions with Applications to Image Databases. ICCV 1998

Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

Which hyperplane is best?

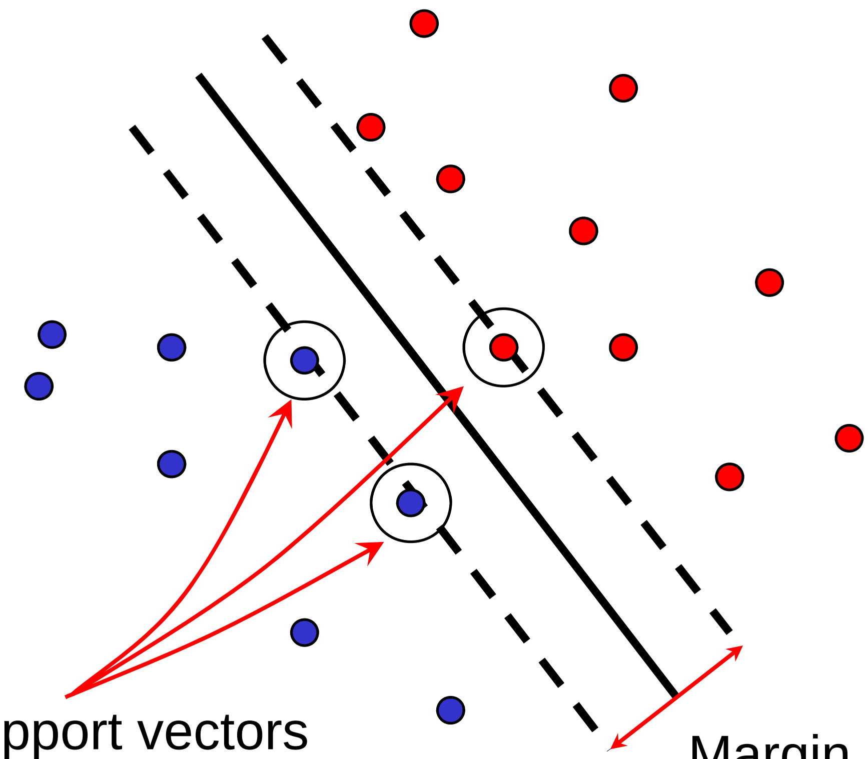
Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples

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

Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support, vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\quad \frac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

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

Finding the maximum margin hyperplane

1. Maximize margin $2/||\mathbf{w}||$
2. Correctly classify all training data:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

Quadratic optimization problem:

Minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

Subject to $y_i(\mathbf{w}\cdot\boldsymbol{x}_i + b) \geq 1$

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

Finding the maximum margin hyperplane

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

learned weight

Support vector

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

Finding the maximum margin hyperplane

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ for any support vector

- Classification function (decision boundary):

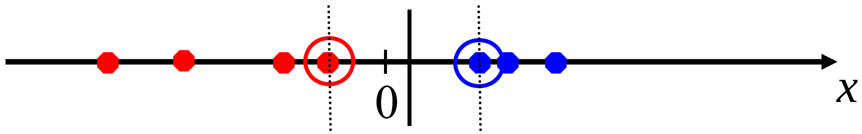
$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$
- Solving the optimization problem also involves computing the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points

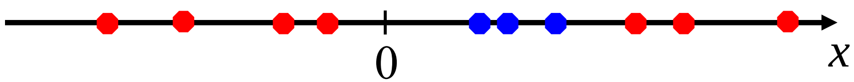
C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Nonlinear SVMs

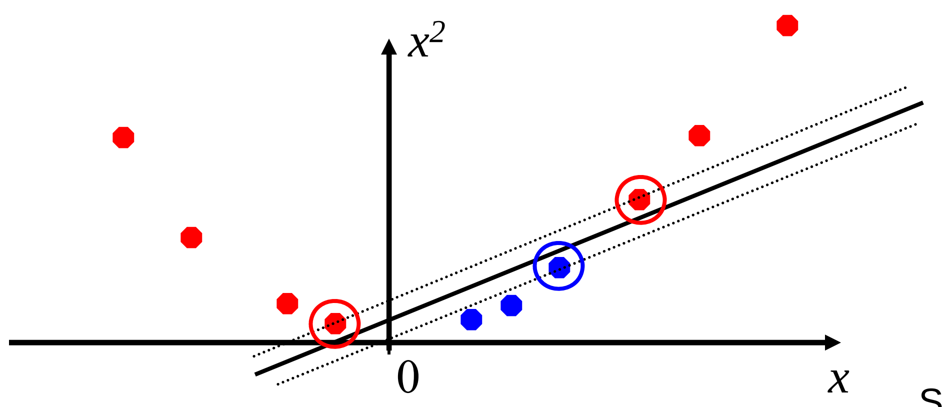
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



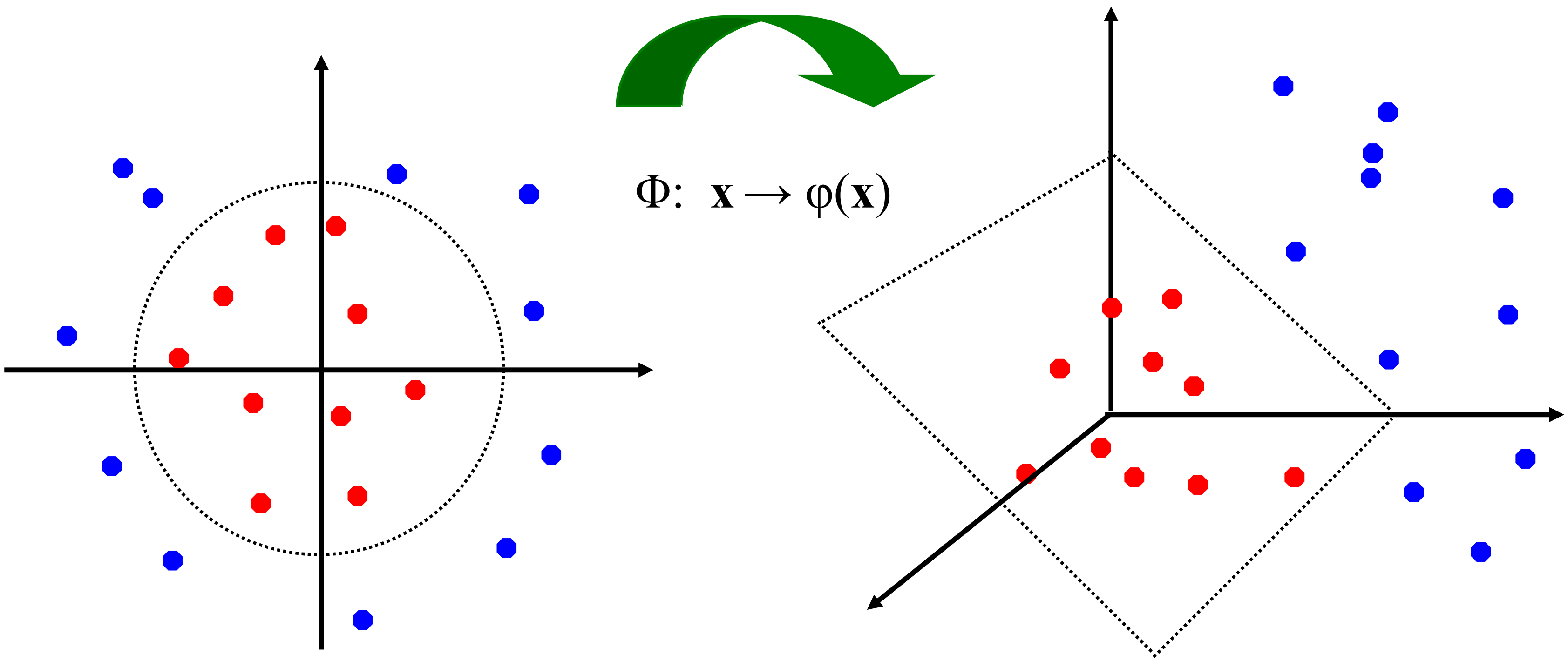
- We can map it to a higher-dimensional space:



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

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:

$$\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

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

Kernels for bags of features

- Histogram intersection kernel:

$$I(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i))$$

- Generalized Gaussian kernel:

$$K(h_1, h_2) = \exp\left(-\frac{1}{A} D(h_1, h_2)^2\right)$$

- *D* can be Euclidean distance, χ^2 distance, Earth Mover's Distance, etc.

J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, Local Features and Kernels for Classifcation of Texture and Object Categories: A Comprehensive Study, IJCV 2007

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

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

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

Summary: Discriminative methods

- Nearest-neighbor and k-nearest-neighbor classifiers
  - L1 distance, χ^2 distance, quadratic distance, Earth Mover's Distance
- Support vector machines
  - Linear classifiers
  - Margin maximization
  - The kernel trick
  - Kernel functions: histogram intersection, generalized Gaussian, pyramid match
  - Multi-class
- Of course, there are many other classifiers out there
  - Neural networks, boosting, decision trees, …