Linear Classification with discriminative models

Jakob Verbeek

December 9, 2011

Course website:

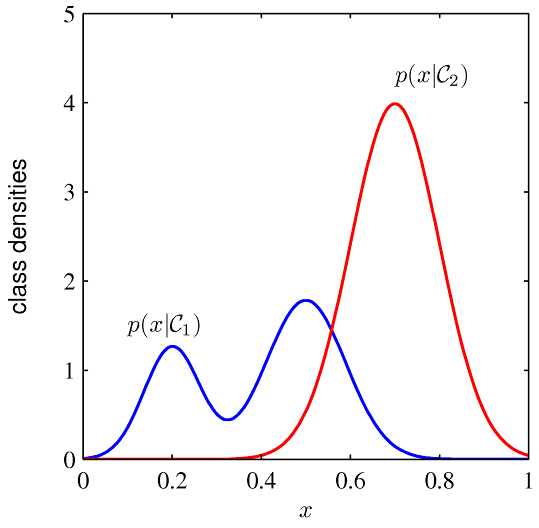
http://lear.inrialpes.fr/~verbeek/MLCR.11.12.php

Generative vs Discriminative Classification

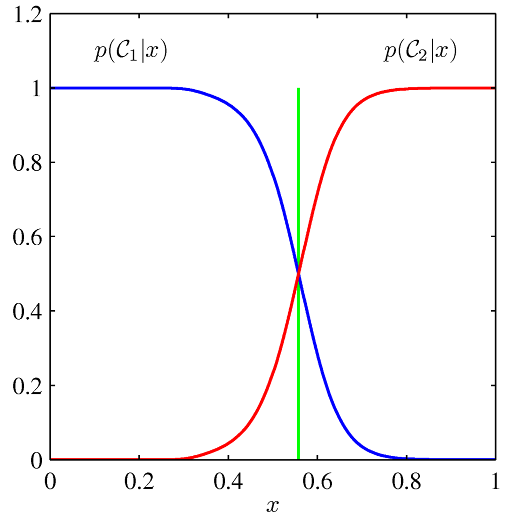
- Training data consists of "inputs", denoted x, and corresponding output "class labels", denoted as y.

- Goal is to predict class label y for a given test data x.

- Generative probabilistic methods
  - Model the density of inputs x from each class p(x|y)
  - Estimate class prior probability p(y)
  - Use Bayes' rule to infer distribution over class given input



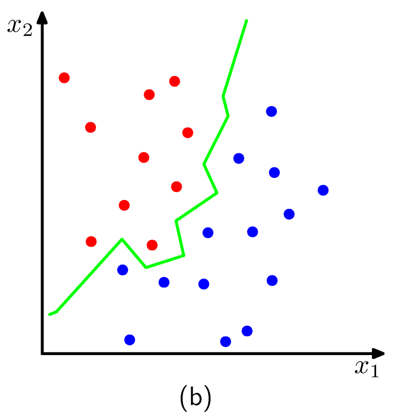
$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)} \qquad p(x) = \sum_y p(y)p(x \mid y)$$



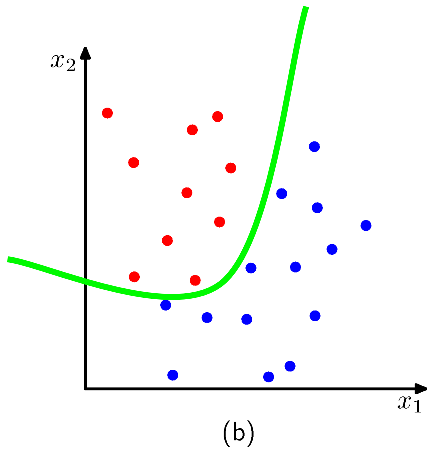
- Discriminative (probabilistic) methods
  - Directly estimate class probability given input: p(y|x)
  - Some methods do not have probabilistic interpretation, but fit a function f(x), and assign to classes based on sign(f(x))

Discriminant function

1. Choose class of decision functions in feature space.
2. Estimate the function parameters from the training set.
3. Classify a new pattern on the basis of this decision rule.



kNN example from last week
Needs to store all data

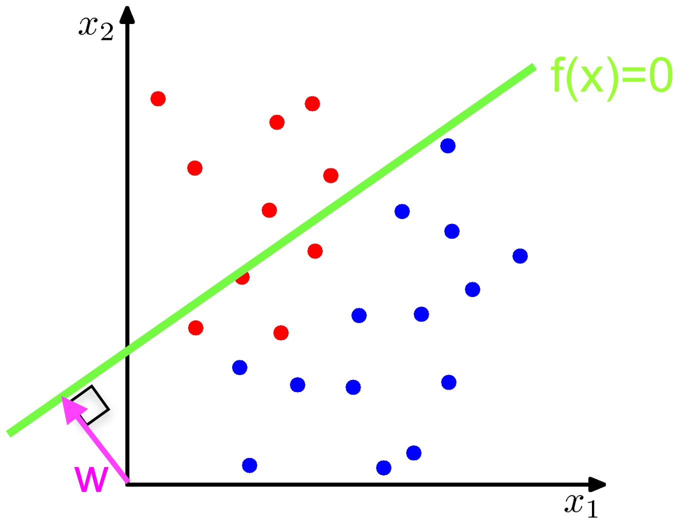


Separation using smooth curve
Only need to store curve parameters

Linear classifiers

- Decision function is linear in the features:

$$f(x) = w^T x + b = b + \sum_{d=1}^{D} w_i x_i$$



- Classification based on the sign of f(x)
- Orientation is determined by ***w*** (surface normal)
- Offset from origin is determined by *b*
- Decision surface is (d-1) dimensional hyper-plane orthogonal to **w**, given by

$$f(x) = w^T x + b = 0$$

Linear classifiers

- Decision function is linear in the features:

$$f(x) = w^T x + b = b + \sum_{d=1}^{D} w_i x_i$$

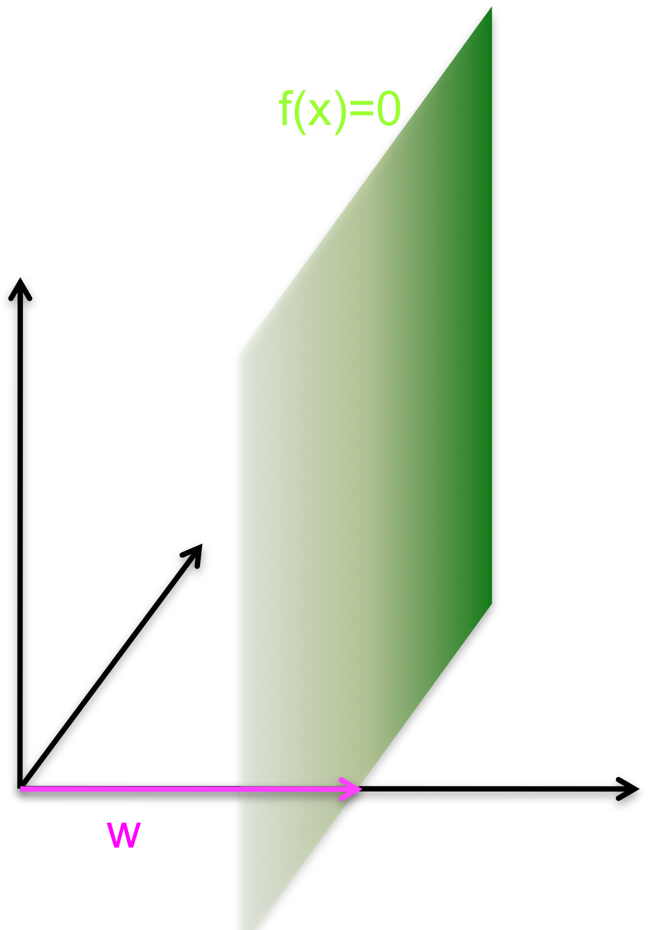
- Classification based on the sign of f(x)

- Orientation is determined by ***w*** (surface normal)
- Offset from origin is determined by *b*

- Decision surface is (d-1) dimensional hyper-plane orthogonal to **w**, given by

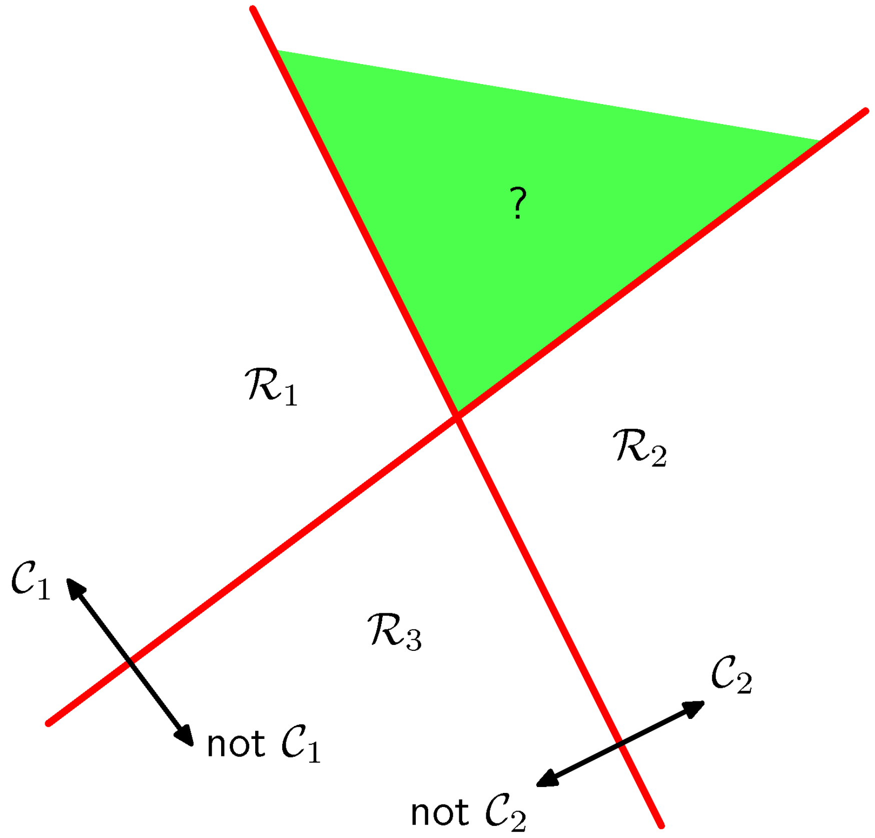
$$f(x) = w^T x + b = 0$$

- What happens in 3d with ***w***=(1,0,0) and *b* = - 1 ?



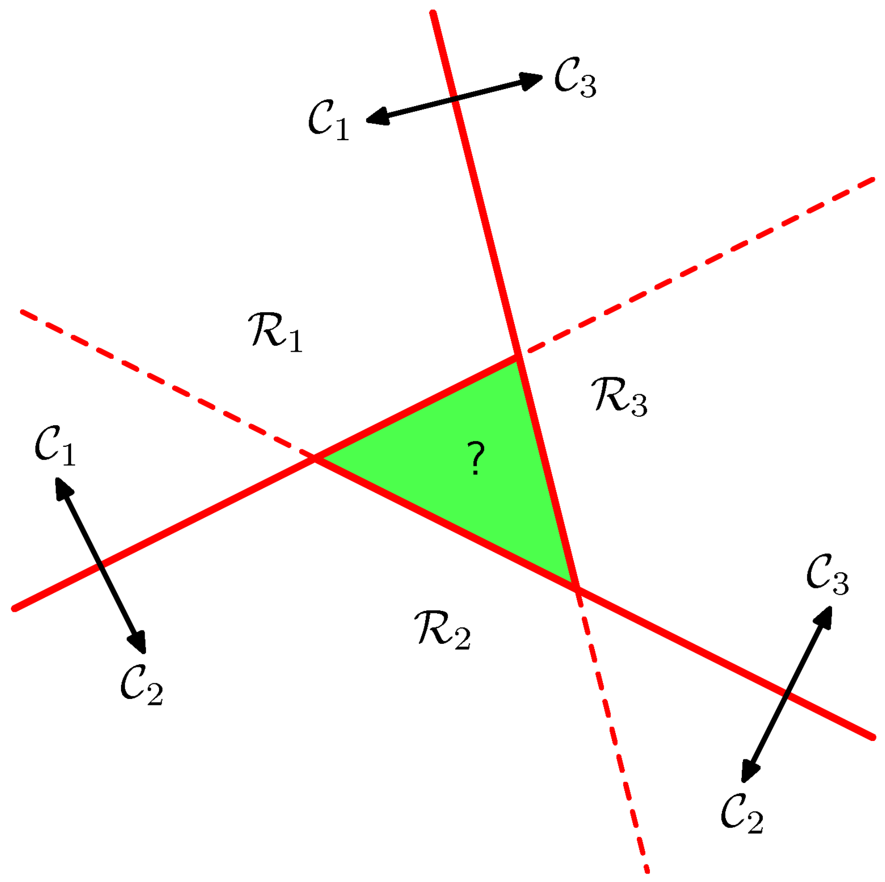
Dealing with more than two classes

- First (bad) idea: construction from multiple binary classifiers
  - Learn the 2-class “base” classifiers independently
  - One vs rest classifiers: train 1 vs (2 & 3), and 2 vs (1 & 3), and 3 vs (1 & 2)
  - Problem: Region claimed by several classes



Dealing with more than two classes

- First (bad) idea: construction from multiple binary classifiers
  - Learn the 2-class "base" classifiers independently
  - One vs One classifiers: train 1 vs 2, and 2 vs 3 and 1 vs 3
  - Problem: conflicts in some regions



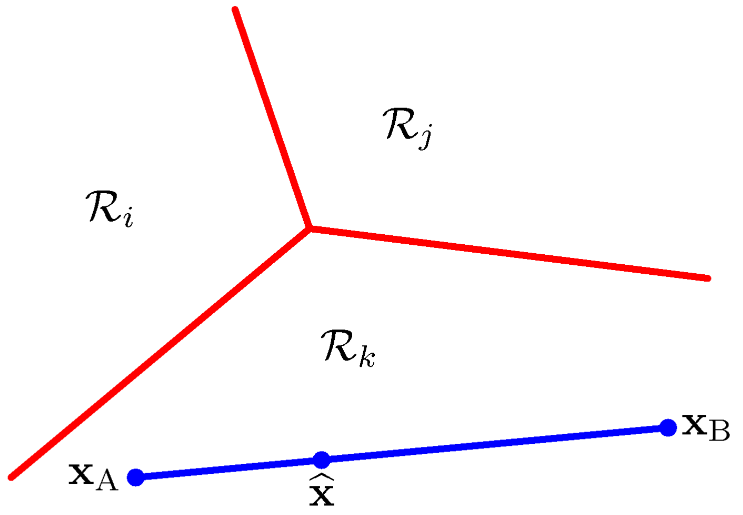
Dealing with more than two classes

- Alternative: define a separate linear function for each class

$$f_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + b_k$$

- Assign sample to the class of the function with maximum value

$$y = \arg\max_k f_k(\mathbf{x})$$



- Decision regions are convex in this case
  - If two points fall in the region, then also all points on connecting line

- What shape do the separation surfaces have ?

Logistic discriminant for two classes

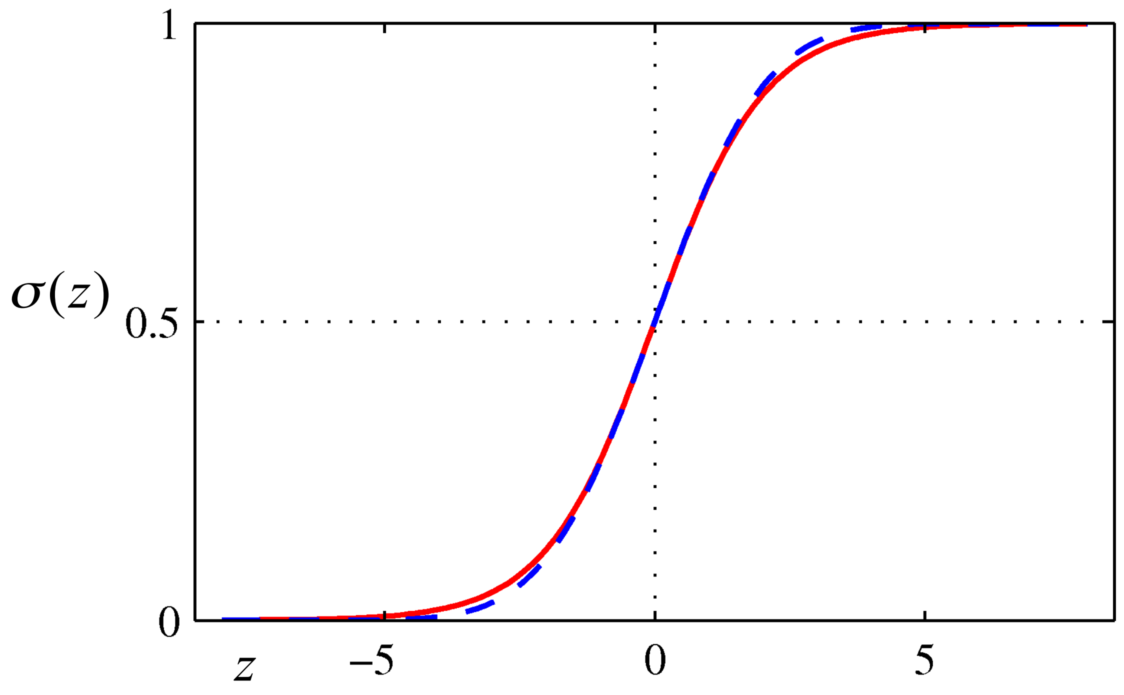
- Map linear score function to class probabilities with sigmoid function

$$f(\mathbf{x}) = b + \mathbf{w}^T \mathbf{x},$$

$$p\left(y = +1 \mid \mathbf{x}\right) = \sigma(f(\mathbf{x})),$$

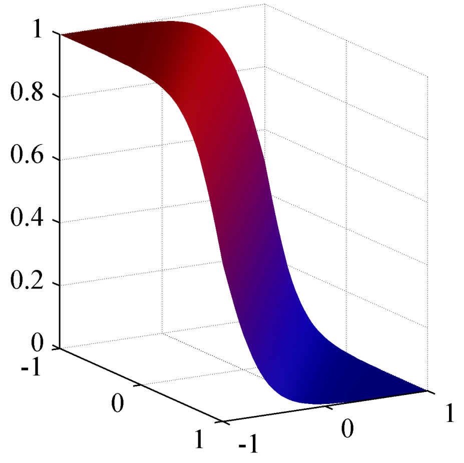
$$\sigma(z) = \frac{1}{1 + \exp\left(-z\right)},$$

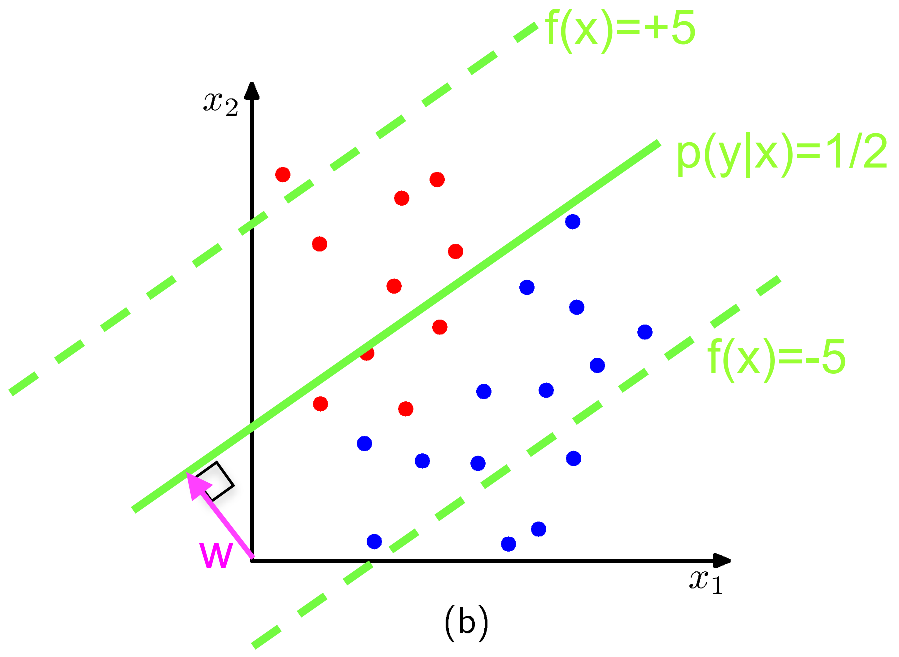
$$p\left(y = -1 \mid \mathbf{x}\right) = 1 - p(y = +1 \mid \mathbf{x}) = \sigma(-f(\mathbf{x}))$$



Logistic discriminant for two classes

- Map linear score function to class probabilities with sigmoid function
- The class boundary is obtained for p(y|x)=1/2, thus by setting linear function in exponent to zero





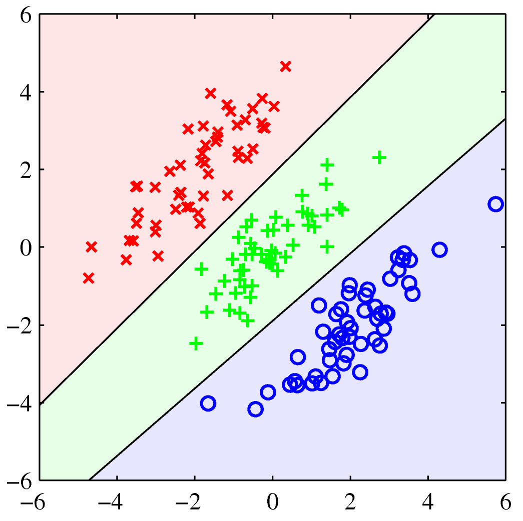
Multi-class logistic discriminant

- Map K linear score functions (one for each class) to K class probabilities using the “soft-max” function

$$f_k(\mathbf{x}) = b_k + \mathbf{w}_k^{\mathbf{T}}\mathbf{x}$$

$$p(y = i \mid \mathbf{x}) = \frac{\exp(f_i(\mathbf{x}))}{\sum_{k=1}^{K}\exp(f_k(\mathbf{x}))}$$

- The class probability estimates are non-negative, and sum to one.
- Probability for the most likely class increases quickly with the difference in the linear score functions
- For any given pair of classes we find that they are equally likely on a hyperplane in the feature space



Parameter estimation for logistic discriminant

- Maximize the (log) likelihood of predicting the correct class label for training data, ie the sum log-likelihood of all training data

$$L = \sum_{n=1}^{N} \log p(y_n \mid \mathbf{x}_n)$$

- Derivative of log-likelihood as intuitive interpretation

$$\frac{\partial L}{\partial b_c} = \sum_n \left([y_n = c] - p(y = c \mid \mathbf{x}_n) \right)$$

Expected number of points from each class should equal the actual number.

Indicator function
1 if y_n=c, else 0

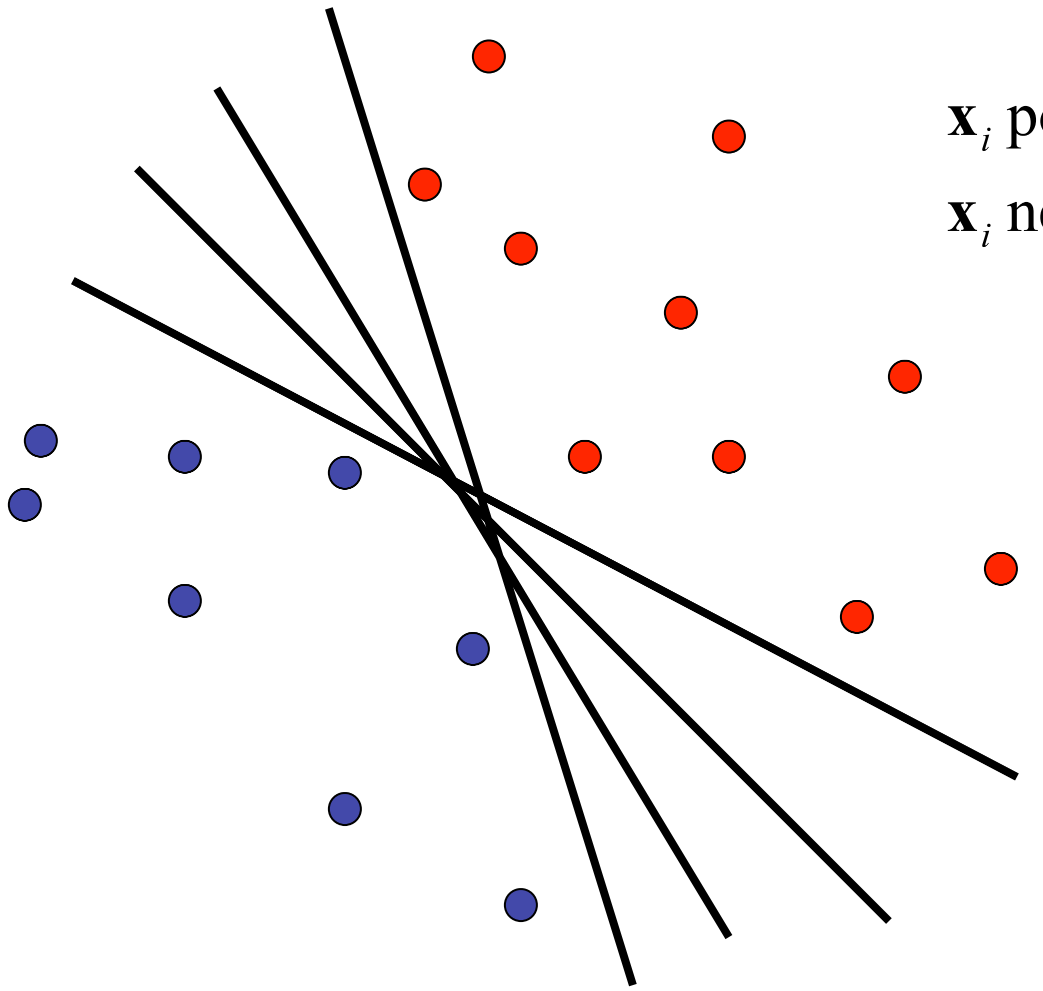
$$\frac{\partial L}{\partial \mathbf{w}_c} = \sum_n \left([y_n = c] - p(y = c \mid \mathbf{x}_n) \right) \mathbf{x}_n = \sum_n a_n \mathbf{x}_n$$

Expected value of each feature, weighting points by p(y|x), should equal empirical expectation.

- No closed-form solution, use gradient-descent methods
  - Note 1: log-likelihood is concave in parameters, hence no local optima
  - Note 2: ***w*** is linear combination of data points

Support Vector Machines

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



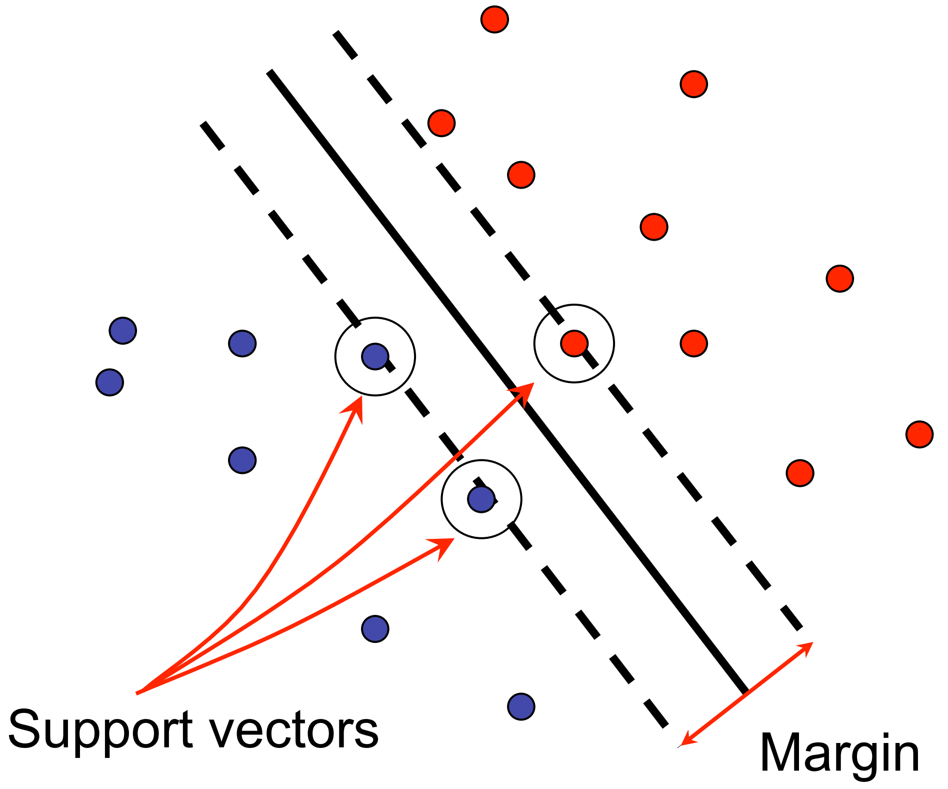
$\mathbf{x}_i$ positive: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

$\mathbf{x}_i$ 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



$\mathbf{x}_i$ 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: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

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

Finding the maximum margin hyperplane

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

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

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

- *Quadratic optimization problem*:
- Minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

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

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):

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

- Notice that it relies on an *inner product* between the test point ***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

Summary Linear discriminant analysis

- Two most widely used linear classifiers in practice:
  - Logistic discriminant (supports more than 2 classes directly)
  - Support vector machines (multi-class extensions recently developed)

- In both cases the weight vector **w** is a linear combination of the data points

$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i$$

- This means that we only need the inner-products between data points to calculate the linear functions

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = b + \sum_{n=1}^{N} \alpha_n \mathbf{x}_n^T \mathbf{x}$$

$$= b + \sum_{n=1}^{N} \alpha_n k(\mathbf{x}_n, \mathbf{x})$$

  - The function k that computes the inner products is called a kernel function

Additional reading material

- Pattern Recognition and Machine Learning
  Chris Bishop, 2006, Springer

- Chapter 4
  - Section 4.1.1 & 4.1.2
  - Section 4.2.1 & 4.2.2
  - Section 4.3.2 & 4.3.4

  - Section 7.1 start + 7.1.1 & 7.1.2