Generative and discriminative classification techniques

Machine Learning and Category Representation 2012-2013

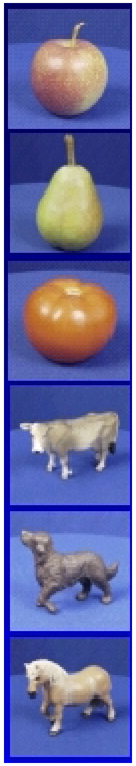
Jakob Verbeek, December 7, 2012

Course website:

http://lear.inrialpes.fr/~verbeek/MLCR.12.13

Classification

Given: training images and their categories



What are the categories of these test images?

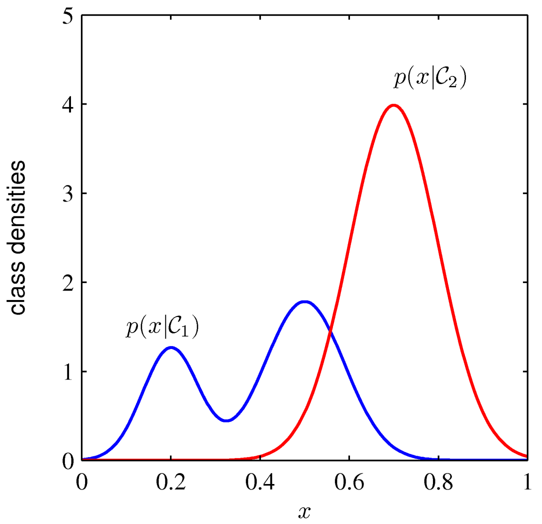
Classification

- Goal is to predict for a test data input the corresponding class label.
  - **Data input x**, eg. image but **could be anything**, format may be vector or other
  - **Class label y**, can take one out of at least 2 **discrete** values, can be more
- In binary classification we often refer to one class as “positive”, and the other as “negative”
- Classifier: function f(x) that assigns a class to x, or probabilities over the classes.
- Training data: pairs (x,y) of inputs x, and corresponding class label y.
- Learning a classifier: determine function f(x) from some family of functions based on the available training data.
- Classifier partitions the input space into regions where data is assigned to a given class
  - Specific form of these boundaries will depend on the family of classifiers used

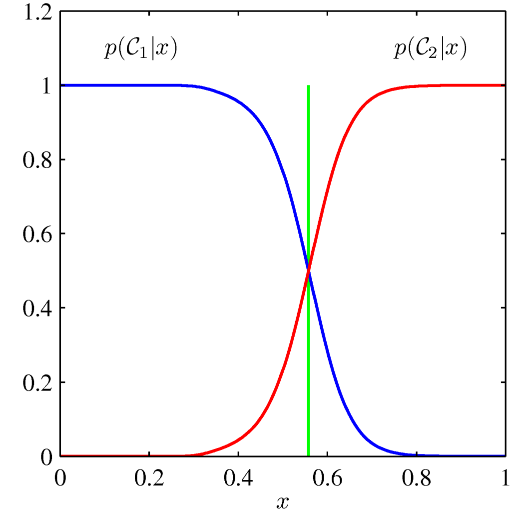
Discriminative vs generative methods

- 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

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



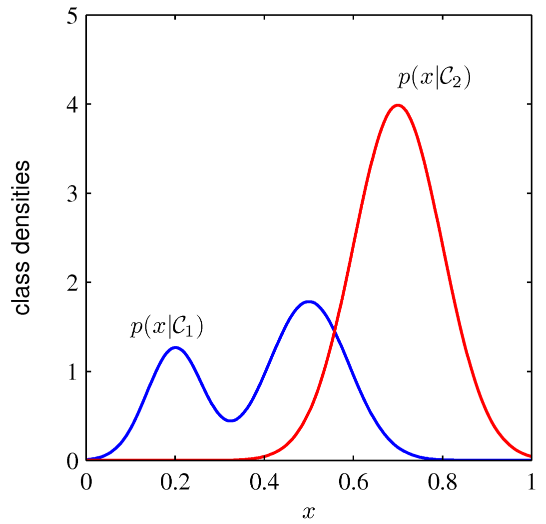
- Discriminative (probabilistic) methods
  - Directly estimate class probability given input: p(y|x)
  - Some methods do not have probabilistic interpretation,
    - eg. they fit a function f(x), and assign to class 1 if f(x)>0, and to class 2 if f(x)<0



Generative classification methods

- 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

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



1. Selection of model class:
   - Parametric model: Gaussian (for continuous), Bernoulli (for binary), …
   - Semi-parametric models: mixtures of Gaussian / Bernoulli / …
   - Non-parametric models: histograms, nearest-neighbor method, …

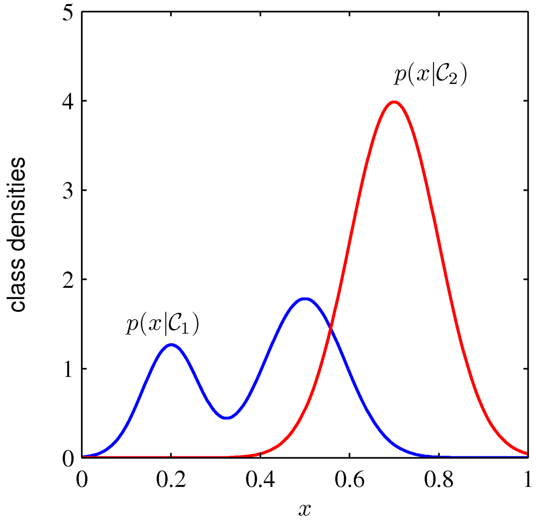
2. Estimate parameters of density for each class to obtain p(x|y)
   - Eg: run EM to learn Gaussian mixture on data of each class

3. Estimate prior probability of each class
   - If data point is equally likely given each class, then assign to the most probable class.
   - Prior probability might be different than the number of available examples !

Generative classification methods

- 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 predict classes given input**

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

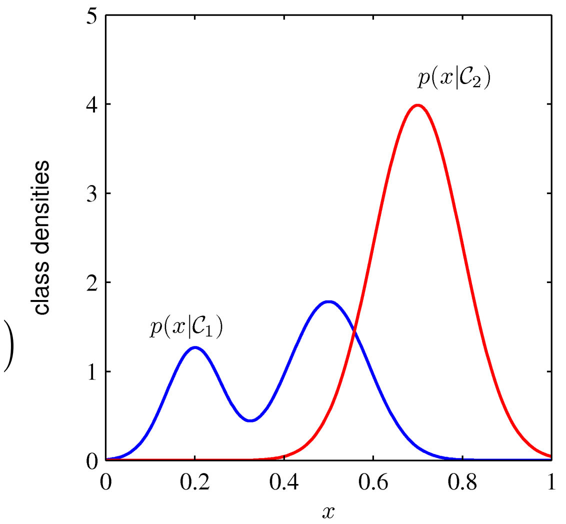


- Given class conditional model, classification is trivial: just apply Bayes' rule
  - Compute p(x|class) for each class,
  - multiply with class prior probability
  - Normalize to obtain the class probabilities

- Adding new classes can be done by adding a new class conditional model
  - Existing class conditional models stay as they are
  - Just estimate p(x|new class) from training examples of new class
  - Plug-in the new class model when using Bayes-rule to predict class

Generative classification methods

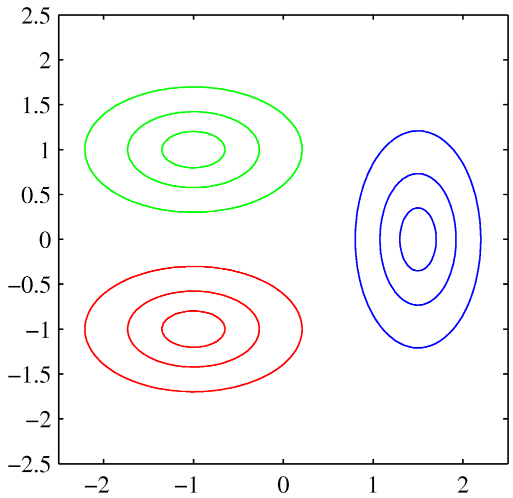
- Generative probabilistic methods
  - Model the density of inputs x from each class p(x|y)
  - Estimate class prior probability p(y)
  - Use Bayes' rule to predict classes given input



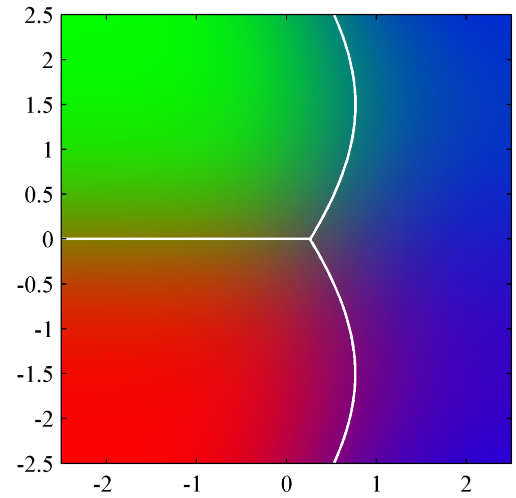
$$p(y|x) = \frac{p(y)\,p(x|y)}{p(x)} \qquad p(x) = \sum_y p(y)\,p(x|y)$$

- **Three-class example in 2d with parametric model**
  - Single Gaussian model per class, equal mixing weights
  - Exercise: characterize the surface of equal class probability when the covariance matrices are equal

p(x|y)

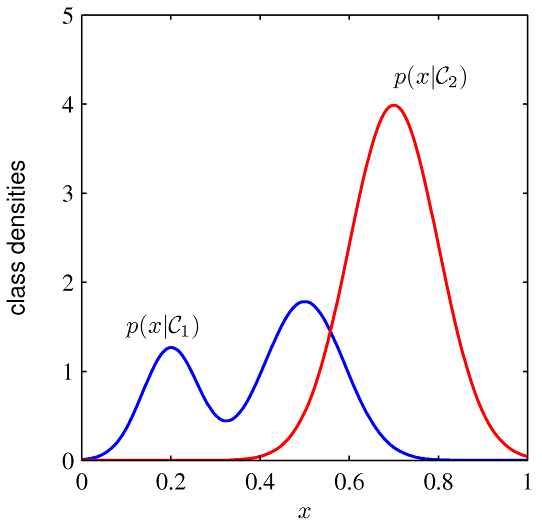


p(y|x)



Generative classification methods

- 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



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

1. Selection of model class:
   - Parametric model: Gaussian (for continuous), Bernoulli (for binary), …
   - Semi-parametric models: mixtures of Gaussian, mixtures of Bernoulli, …
   - **Non-parametric models: histograms, nearest-neighbor method, …**

1. Estimate parameters of density for each class to obtain p(x|class)
   - Eg: run EM to learn Gaussian mixture on data of each class

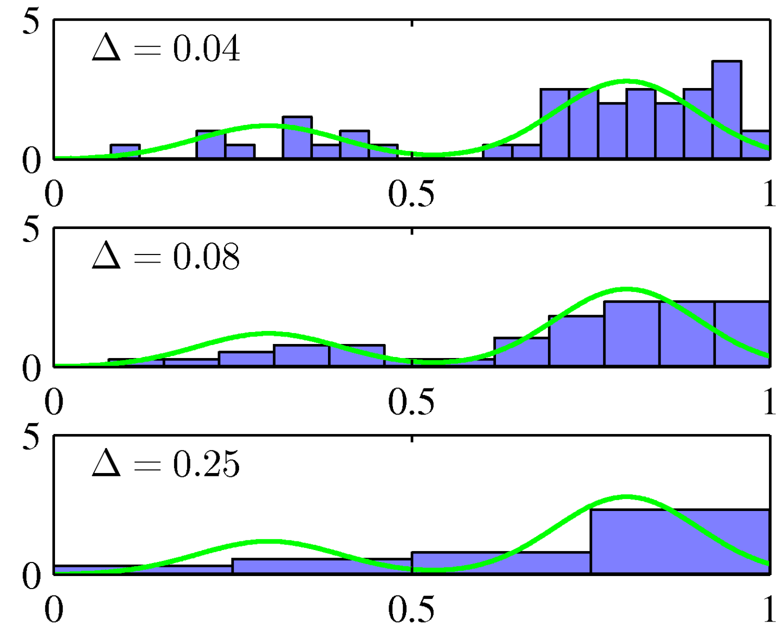
1. Estimate prior probability of each class
   - If data point is equally likely given each class, then assign to the most probable class.
   - Prior probability might be different than the number of available examples !

Histogram density estimation

- Suppose we
  - have N data points
  - use a histogram with C cells
- How to set the density level in each cell ?
  - Maximum likelihood estimator.
  - Proportional to nr of points n in cell
  - Inversely proportional to volume V of cell

$$p_c = \frac{n_c}{N V_c}$$

  - Exercise: derive this result



- Problems with histogram method:
  - **# cells scales exponentially with the dimension of the data**
  - Discontinuous density estimate
  - How to choose cell size?

informatics mathematics

Grenoble INP Ensimag

The ‘curse of dimensionality’

- Number of bins increases exponentially with the dimensionality of the data.
  - Fine division of each dimension: many empty bins
  - Rough division of each dimension: poor density model

- The number of parameters may be reduced by assuming independence between the dimensions of ***x***: the **naïve Bayes model**

$$p(x) = \prod_{d=1}^{D} p(x^d)$$

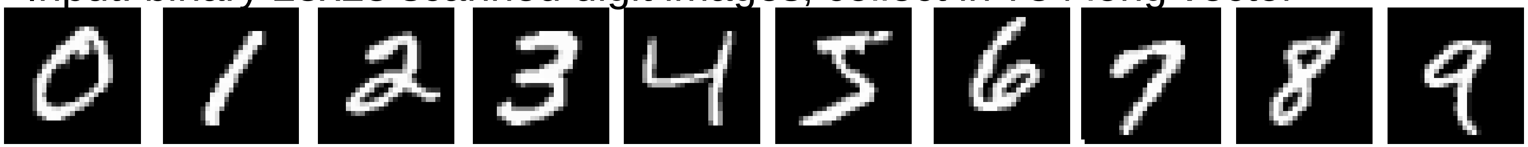
  - For example, for histogram model: we estimate a histogram per dimension
  - Still C^D cells, but only D x C parameters to estimate, instead of C^D

- Model is “naïve” since it assumes that all variables are independent…
  - Unrealistic for high dimensional data, where variables tend to be dependent
  - Typically poor density estimator for p(x|y)
  - Classification performance may still be good using the derived p(y|x)

- Also applies to other distributions, eg multivariate Gaussian, instead of full covariance matrix with D^2 parameters, we estimate variance per dimension

Example of a naïve Bayes model

- Hand-written digit classification
  - Input: binary 28x28 scanned digit images, collect in 784 long vector



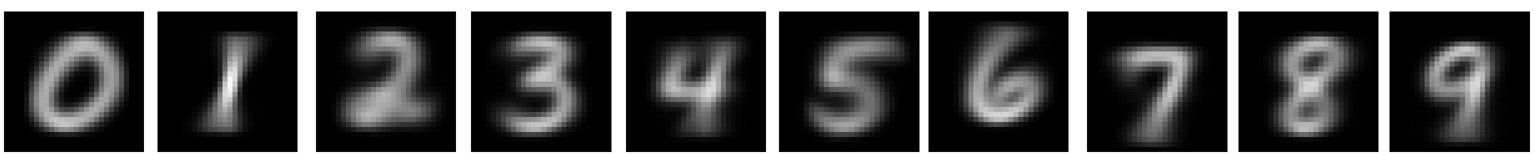
  - Desired output: class label of image

- Generative model over 28 x 28 pixel images (2^{784} possible images)
  - Independent Bernoulli model for each class
  - Probability per pixel per class
  - Maximum likelihood estimator is average value per pixel per class

$$p(x|y=c)=\prod_d p(x^d|y=c)$$

$$p(x^d=1|y=c)=\theta_{cd}$$

$$p(x^d=0|y=c)=1-\theta_{cd}$$



- Classify using Bayes' rule: $p(y|x)=\frac{p(y)\,p(x|y)}{p(x)}$

k-nearest-neighbor density estimation

- Idea: put a cell around the test sample we want to know p(x) for
  - fix number of samples in the cell, find the right cell size.

- Probability to find a point in a sphere ***A*** centered on $\boldsymbol{x_0}$ with volume ***v*** is

$$P(x \in A) = \int_A p(x)\,dx$$

- A smooth density is approximately constant in small region, and thus

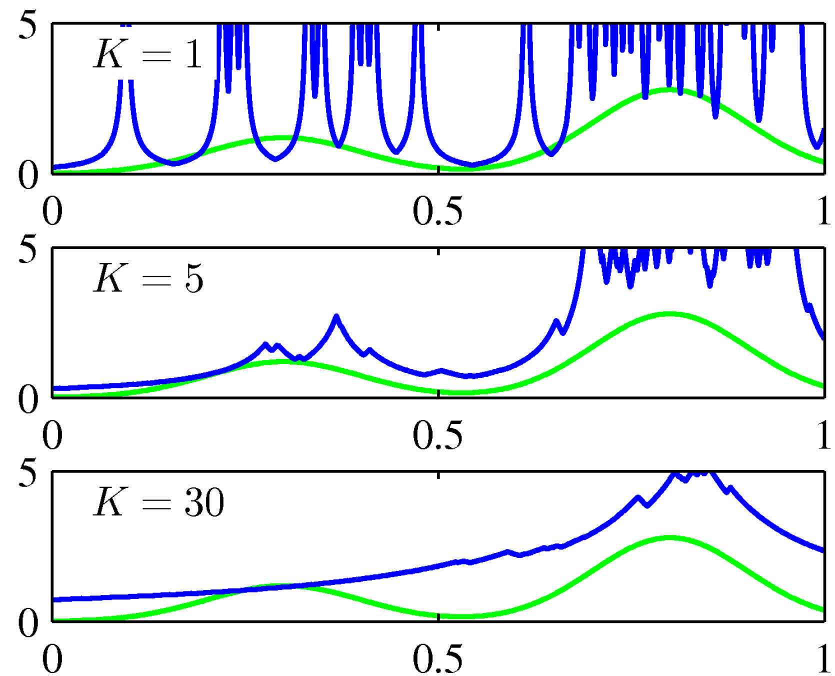
$$P(x \in A) = \int_A p(x)\,dx \approx v\,p(x_0)$$

- Alternatively: estimate ***P*** from the fraction of training data in ***A*** $\quad P(x \in A) \approx \frac{k}{N}$
  - Total N data points, k in the sphere ***A***

- Combine the above to obtain estimate $\quad p(x_0) \approx \frac{k}{Nv}$
  - Density estimates not guaranteed to integrate to one!

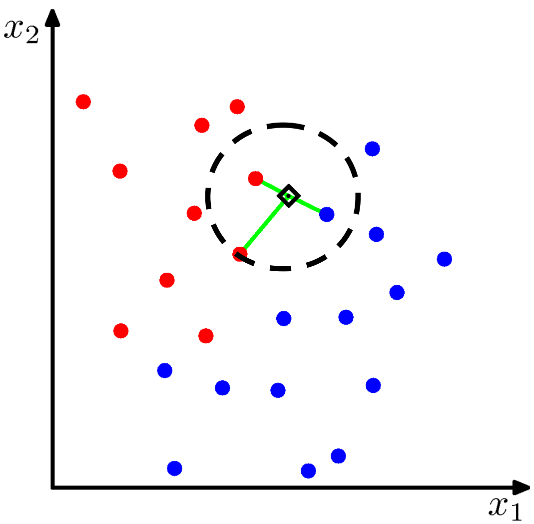
k-nearest-neighbor density estimation

- Procedure in practice:
  - Choose ***k***
  - For given ***x***, compute the volume ***v*** which contain ***k*** samples.
  - Estimate density with $p(x) \approx \frac{k}{Nv}$
- Volume of a sphere with radius ***r*** in ***d*** dimensions is $v(r,d) = \frac{2r^d \pi^{d/2}}{\Gamma(d/2+1)}$
- What effect does ***k*** have?
  - Data sampled from mixture of Gaussians plotted in green
  - Larger ***k***, larger region, smoother estimate
- Selection of k typically by cross validation



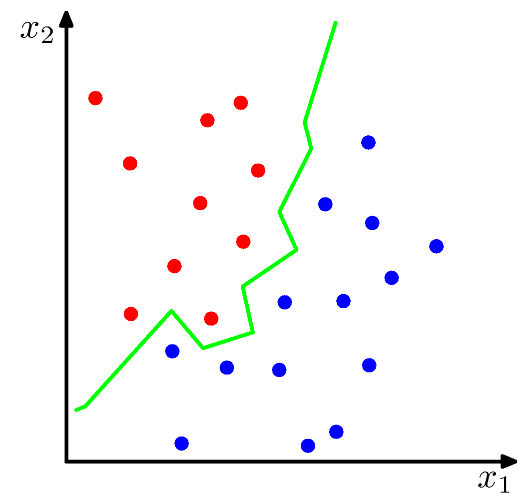
k-nearest-neighbor classification

- Use *k*-nearest neighbor density estimation to find p(x|y)
- Apply Bayes rule for classification: *k*-nearest neighbor classification
  - Find sphere volume v to capture ***k*** data points for estimate
  - Use the same sphere for each class for estimates $p(x|y=c)=\frac{k_c}{N_c v}$
  - Estimate class prior probabilities $p(y=c)=\frac{N_c}{N}$
  - Calculate class posterior distribution



(a)

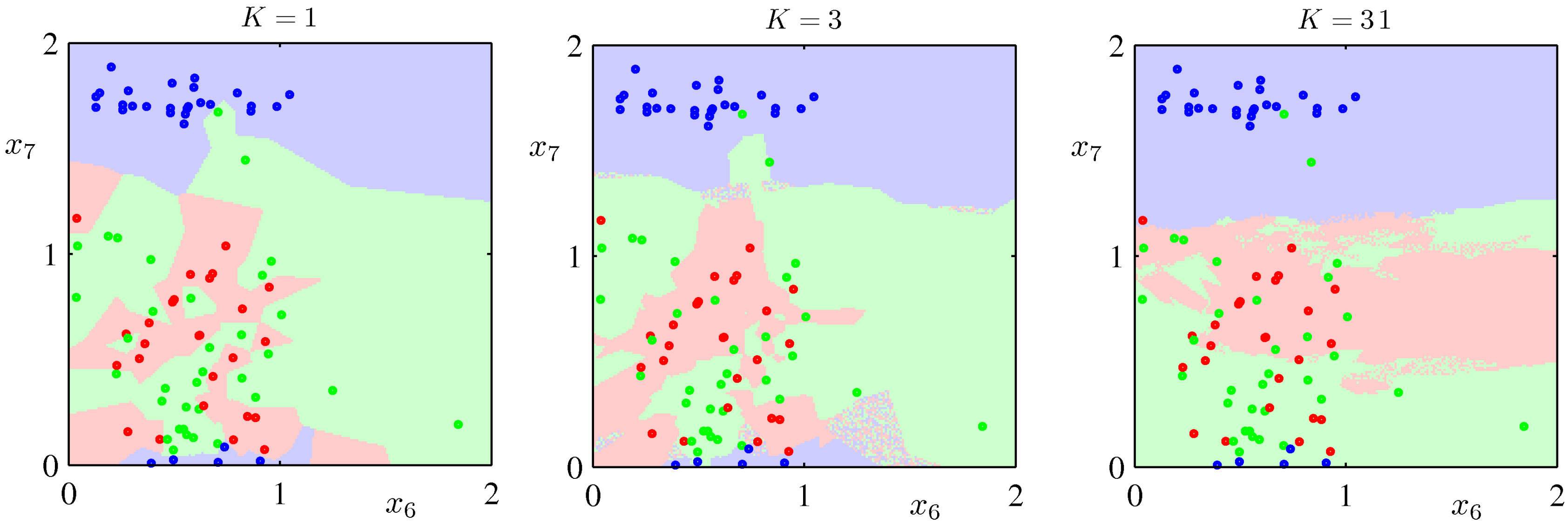
$$p(y=c|x)=\frac{p(y=c)\,p(x|y=c)}{p(x)}$$
$$=\frac{1}{p(x)}\frac{k_c}{Nv}$$
$$=\frac{k_c}{k}$$



(b)

k-nearest-neighbor classification rule

- Effect of k on classification boundary
  - Larger number of neighbors: Larger regions, smoother class boundaries



- Pros: Very simple
  - just set k, and choose a distance measure, no learning
  - Generic: applies to almost anything, as long as you have a distance
- Cons: Very costly when having large training data set
  - Need to store all data (memory)
  - Need to compute distances to all data (time)

Summary generative classification methods

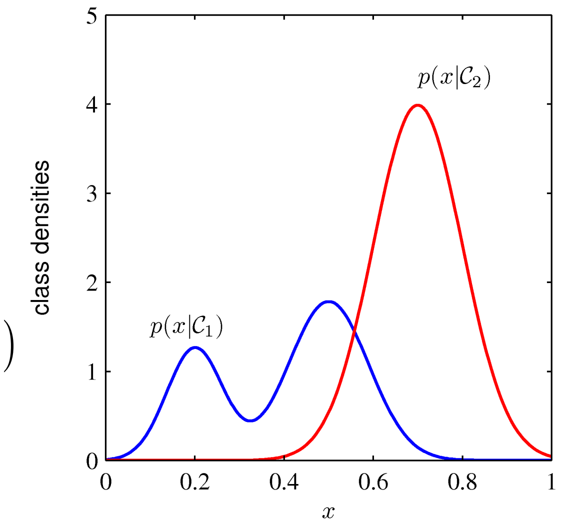
- (Semi-) Parametric models, eg p(x|y) is Gaussian, or mixture of …
  - Pros: no need to store training data, just the class conditional models
  - Cons: may fit the data poorly, and might therefore lead to poor classification result

- Non-parametric models:
  - Advantage is their flexibility: no assumption on shape of data distribution
  - Histograms:
    - Only practical in low dimensional space (<5 or so), application in high dimensional space will lead to exponentially many cells, most of which will be empty
    - Naïve Bayes modeling in higher dimensional cases
  - K-nearest neighbor density estimation: simple but expensive at test time
    - storing all training data (memory space)
    - Computing nearest neighbors (computation)

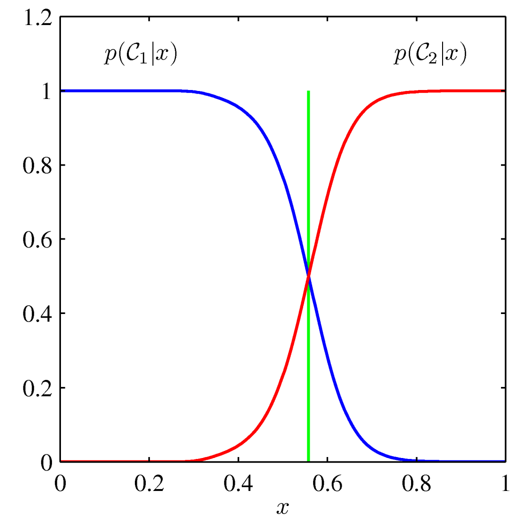
Discriminative vs generative methods

- 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

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

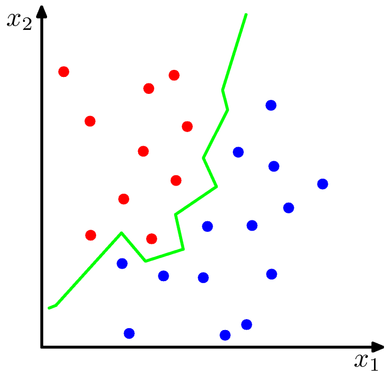


- **Discriminative (probabilistic) methods**
  - Directly estimate class probability given input: p(y|x)
  - Some methods do not have probabilistic interpretation,
    - eg. they fit a function f(x), and assign to class 1 if f(x)>0, and to class 2 if f(x)<0



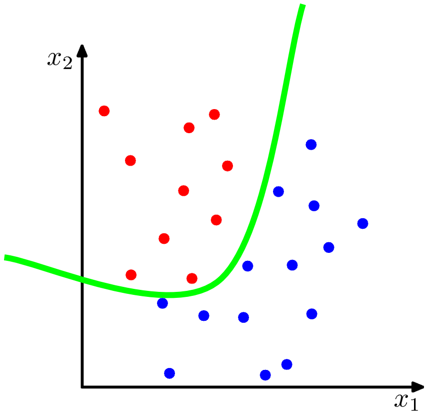
Discriminant function

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.



(b)

kNN classification
Needs to store all data



(b)

Separation using smooth curve
Only need to store curve parameters

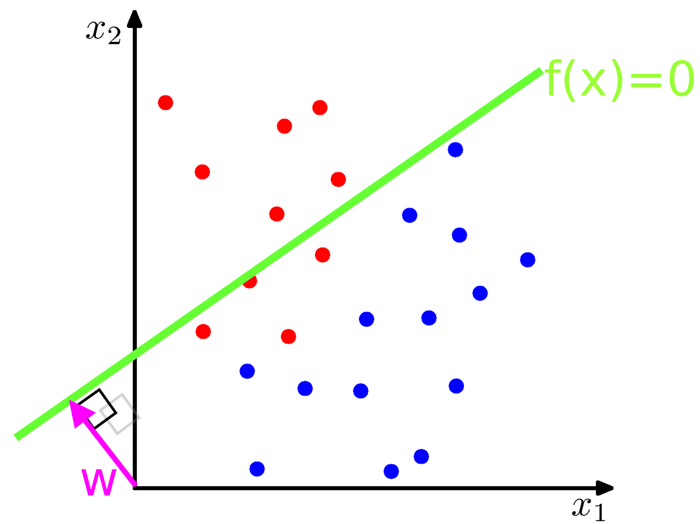
Linear classifiers

- Decision function is linear in the features:

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

- Classification based on the sign of f(x)

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



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

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

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

Linear classifiers

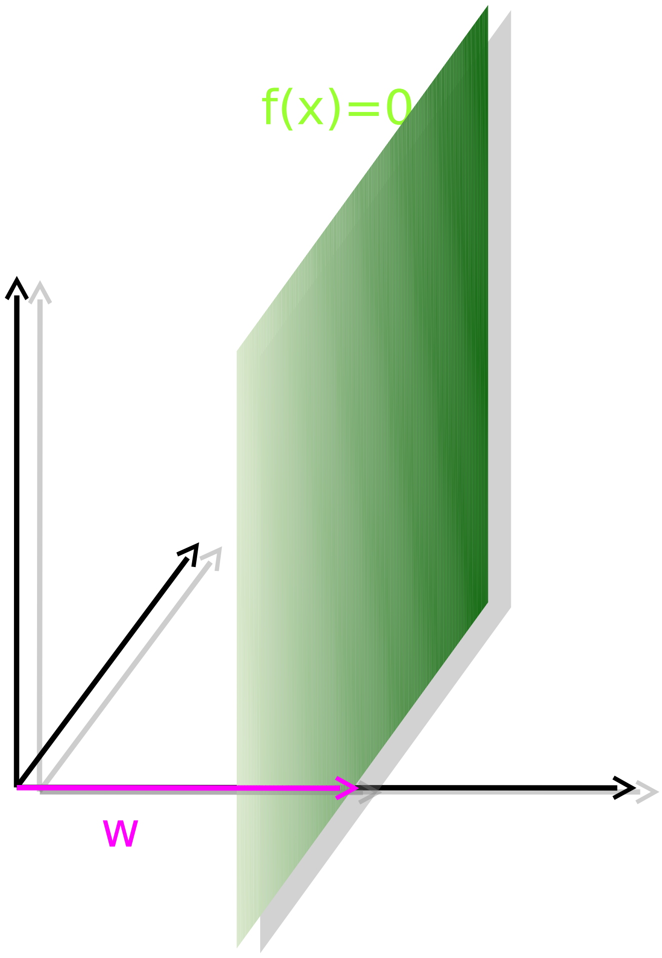
- Decision surface for w=(1,0,0) and b = -1

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

$$f(x) = w^T x + b = b + \sum_{i=1}^{d} w_i x_i = 0$$

$$x_1 - 1 = 0$$

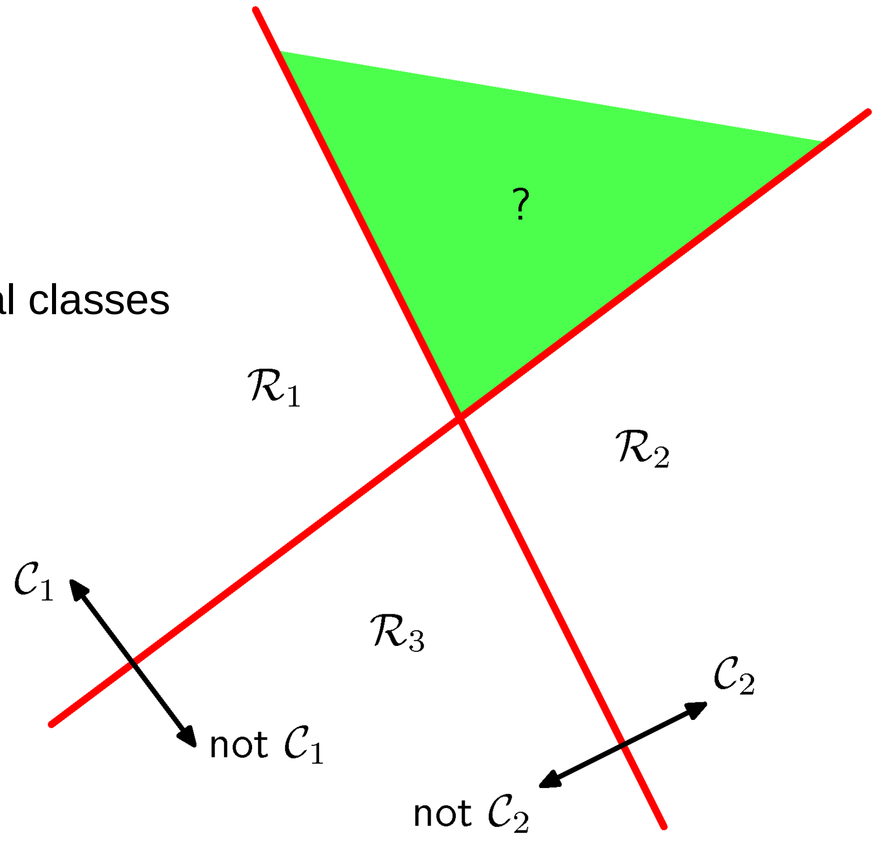
$$x_1 = 1$$



Dealing with more than two classes

- First idea: construction from multiple binary classifiers
  - Learn binary “base” classifiers independently

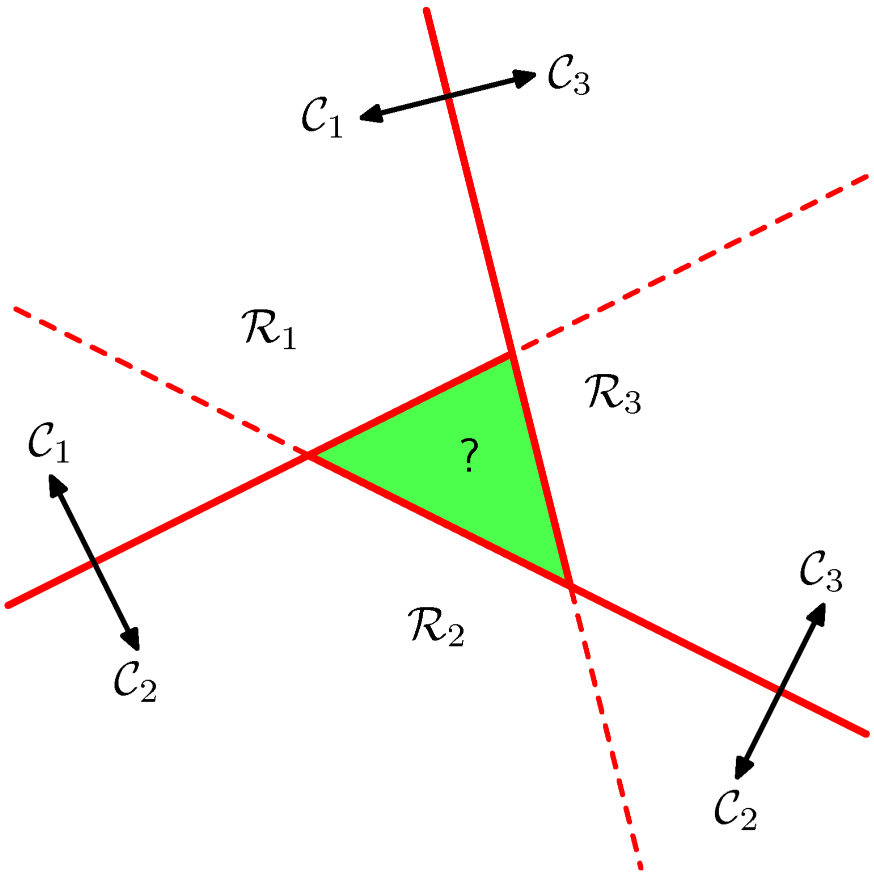
- One vs rest approach:
  - 1 vs (2 & 3)
  - 2 vs (1 & 3)
  - 3 vs (1 & 2)
- Problem: Region claimed by several classes



Dealing with more than two classes

- First idea: construction from multiple binary classifiers
  - Learn binary “base” classifiers independently

- One vs one approach:
  - 1 vs 2
  - 1 vs 3
  - 2 vs 3
- Problem: conflicts in some regions



informatics mathematics

Grenoble INP Ensimag

Dealing with more than two classes

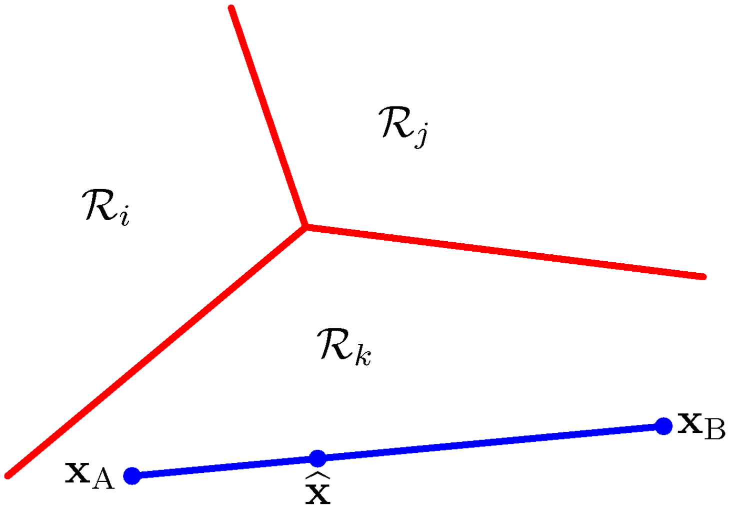
- Alternative: define a separate linear score function for each class

$$f_k(x) = w_k^T x + b_k$$

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

$$y = arg\,max_k\, f_k(x)$$

- Exercise 1: give the expression for points where two classes have equal score



- Exercise 2: show that the set of points assigned to a class is convex
  - If two points fall in the region, then also all points on connecting line

Logistic discriminant for two classes

- Map linear score function to class probabilities with sigmoid function

$$p(y=+1|x)=\sigma(w^T x+b)$$

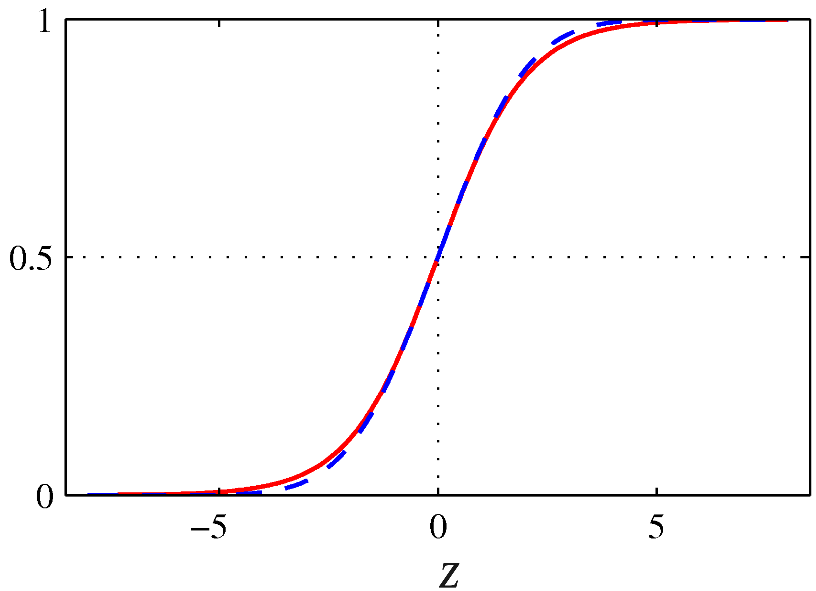
- For binary classification problem, we have by definition

$$p(y=-1|x)=1-p(y=+1|x)$$

- Exercise: show that

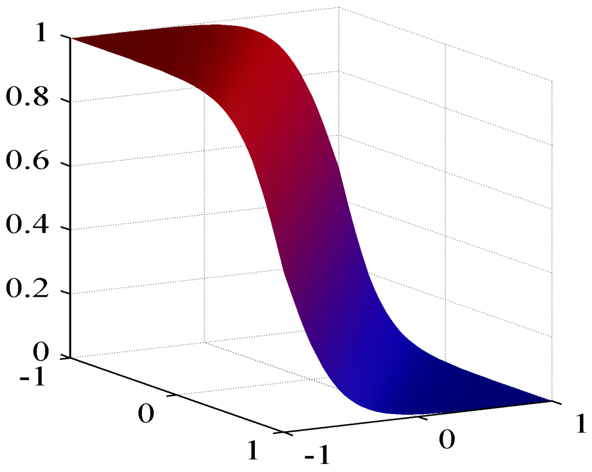
$$p(y=-1|x)=\sigma(-(w^T x+b))$$

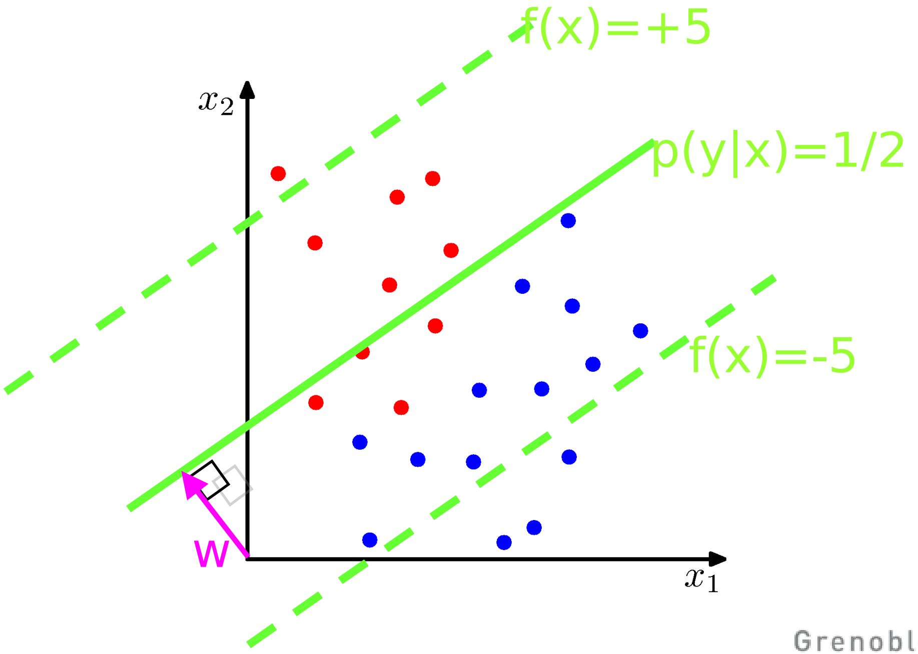
$$\sigma(z)=\frac{1}{1+\exp(-z)}$$



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

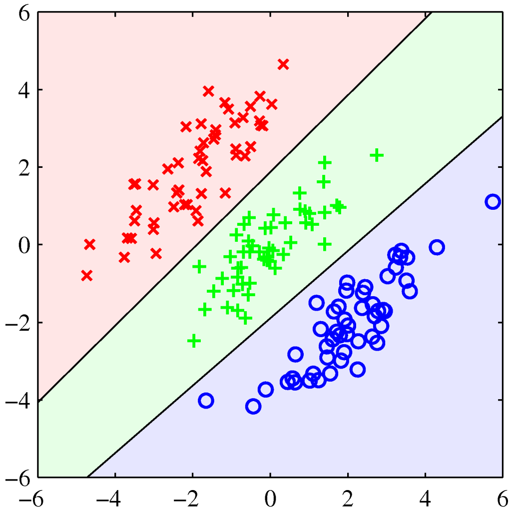
- Map score function of each class to class probabilities with "soft-max" function

$$f_k(x) = w_k^T x + b_k \qquad p(y=c|x) = \frac{\exp(f_c(x))}{\sum_{k=1}^{K} \exp(f_k(x))}$$

  - The class probability estimates are non-negative, and sum to one.
  - Relative probability of most likely class increases exponentially with the difference in the linear score functions

$$\frac{p(y=c|x)}{p(y=k|x)} = \frac{\exp(f_c(x))}{\exp(f_k(x))} = \exp(f_c(x) - f_k(x))$$

  - 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, i.e. the sum log-likelihood of all training data

$$L = \sum_{n=1}^{N} \log p(y_n | x_n)$$

- Derivative of log-likelihood as intuitive interpretation

$$\frac{\partial L}{\partial b_k} = \sum_{n=1}^{N} [y_n = k] - p(y = k | x_n)$$

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

Indicator function
1 if y_n=k, else 0

$$\frac{\partial L}{\partial w_k} = \sum_{n=1}^{N} ([y_n = k] - p(y = k | x_n)) x_n = \sum_{n=1}^{N} \alpha_n x_n$$

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

informatics mathematics

Grenoble INP Ensimag