Generative and discriminative classification techniques

Machine Learning and Category Representation 2013-2014

Jakob Verbeek, December 13+20, 2013

Course website:

http://lear.inrialpes.fr/~verbeek/MLCR.13.14

Classification

Given: training images and their categories

?

To which category does a new image belong?

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.

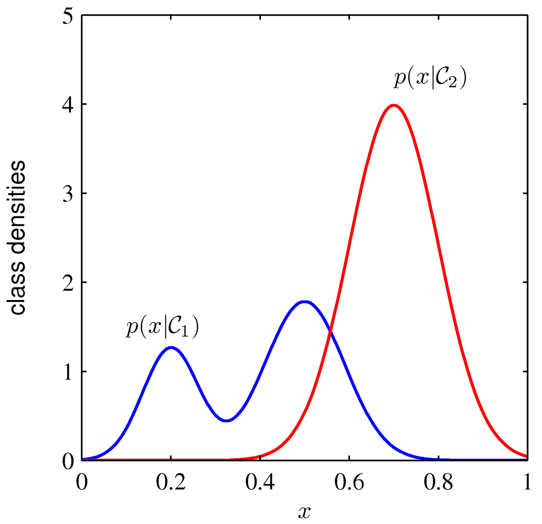
- Learning a classifier: determine function f(x) from some family of functions based on the available training data.

- 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

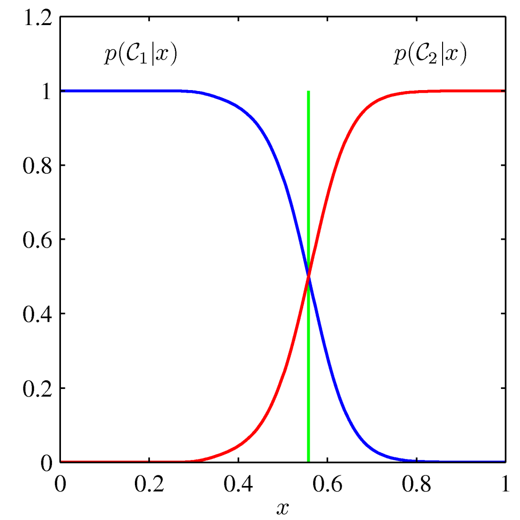
Discriminative vs generative methods

- 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

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

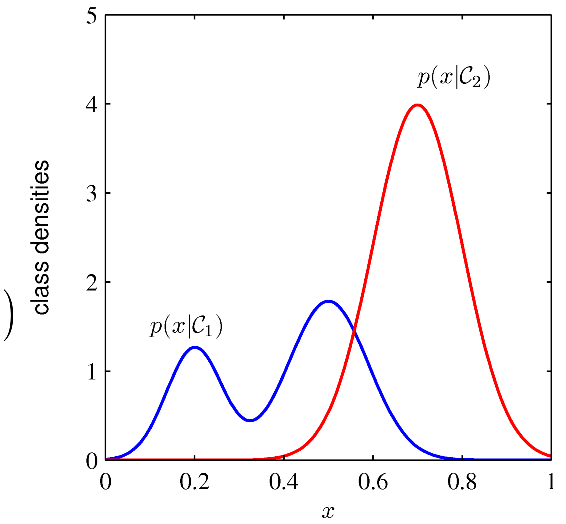


- Discriminative (probabilistic) methods
  - Directly estimate class probability given input: p(y|x)
  - Some methods do not have probabilistic interpretation,
    - eg. they fit a function f(x), and assign to class 1 if f(x)>0, and to class 2 if f(x)<0



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



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

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, …

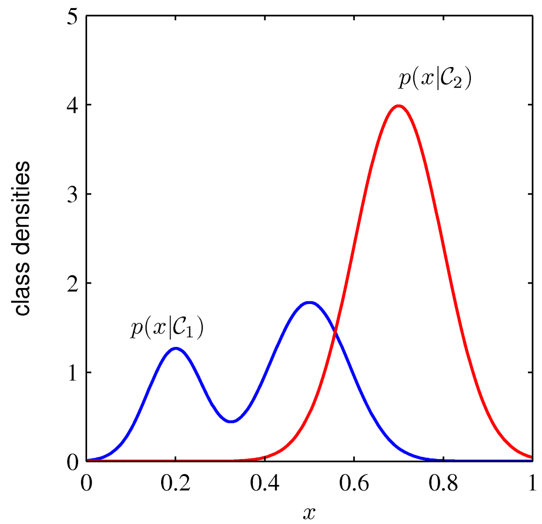
2. Estimate parameters of density for each class to obtain p(x|y)
   - Eg: run EM to learn Gaussian mixture on data of each class

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

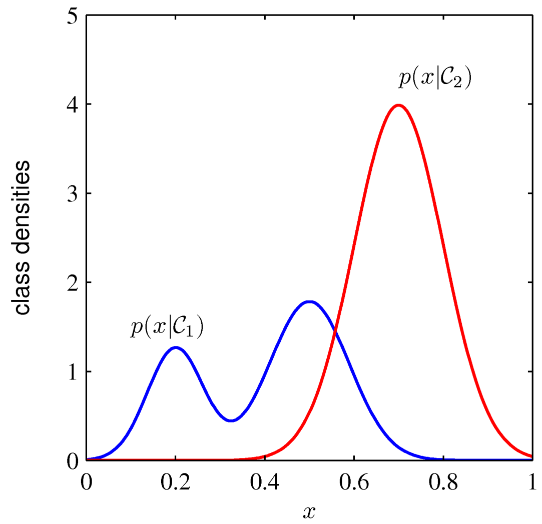
- Adding new classes can be done by adding a new class conditional model
  - Existing class conditional models stay as they are
  - Estimate p(x|new class) from training examples of new class
  - Re-estimate class prior probabilities

informatics mathematics Inria

Grenoble INP Ensimag

Generative classification methods

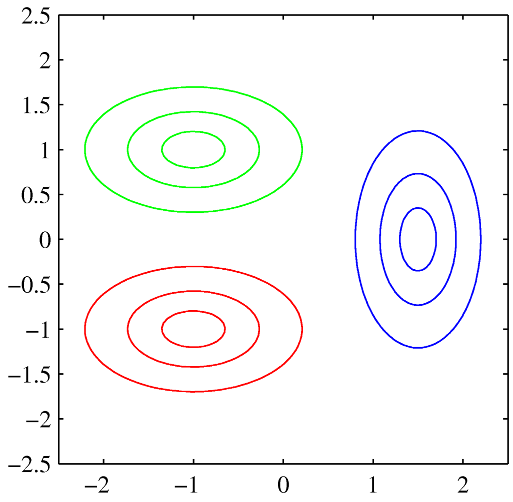
- Generative probabilistic methods
  - Model the density of inputs x from each class p(x|y)
  - Estimate class prior probability p(y)
  - Use Bayes' rule to predict classes given input



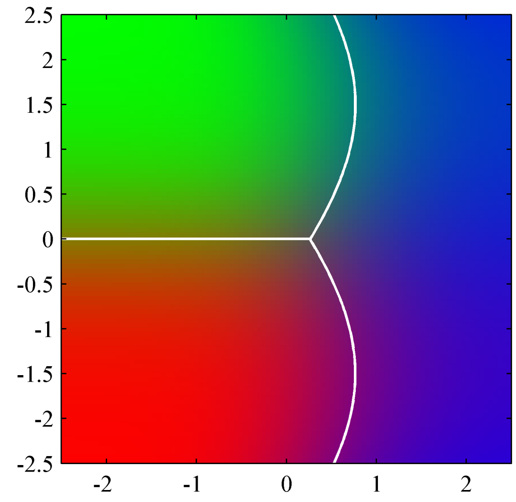
$$p(y|x)=\frac{p(y)\,p(x|y)}{p(x)} \qquad\qquad p(x)=\sum_y p(y)\,p(x|y)$$

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

p(x|y)

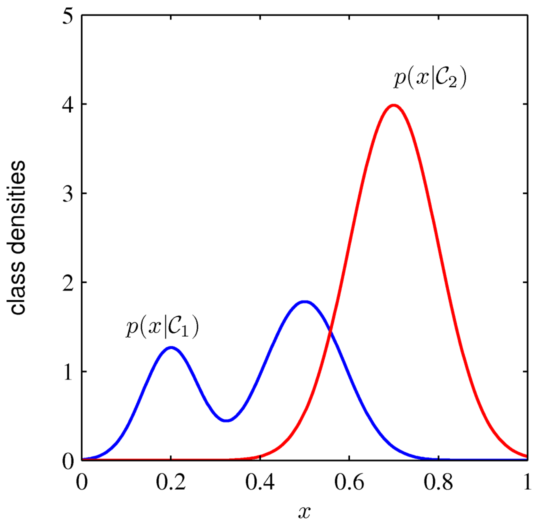


p(y|x)



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



1. Selection of model class:
   - Parametric model: Gaussian (for continuous), Bernoulli (fo
   - 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

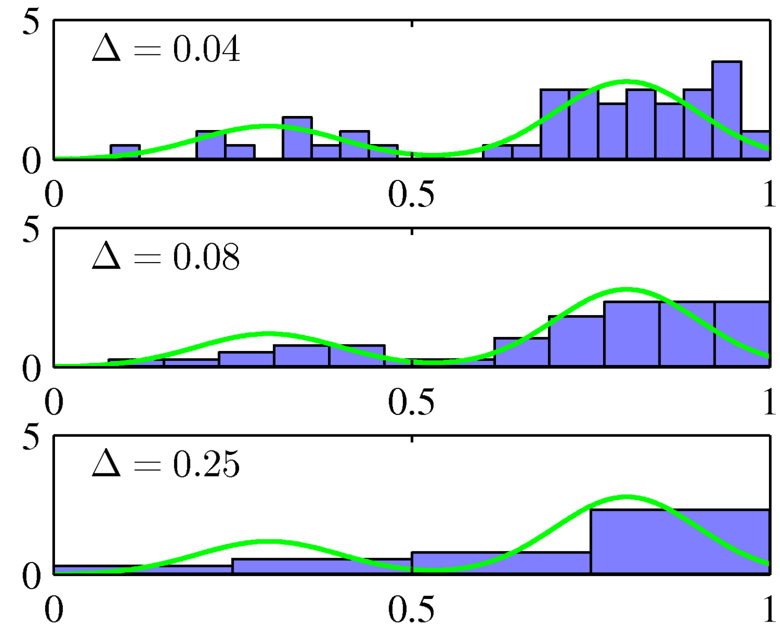
1. Estimate prior probability of each class
   - Fraction of points in training data for each class
   - Assumes class proportions in train data are representative for test time (not always true)

informatics mathematics Inria

Grenoble INP Ensimag

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

$$p_c = \frac{n_c}{NV_c}$$

  - 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?

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)

- Principle can be applied to estimation with any type of model

k-nearest-neighbor density estimation

- Instead of having fixed cells as in histogram method, 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

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

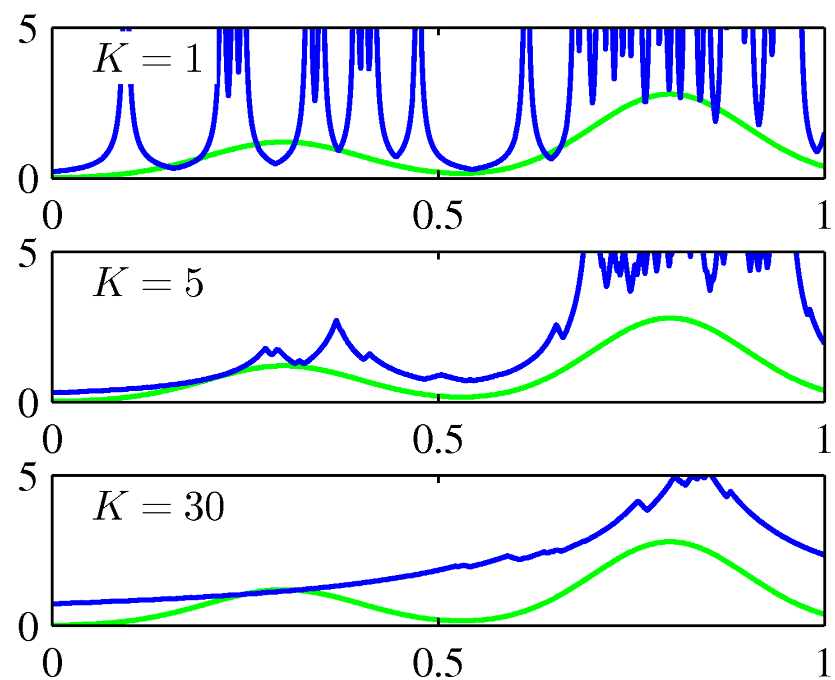
- Alternatively: estimate ***P*** from the fraction of training data in ***A***
  - Total N data points, k in the sphere ***A***

$$P(x \in A) \approx \frac{k}{N}$$

- Combine the above to obtain estimate $p(x_0) \approx \dfrac{k}{Nv}$
  - 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 $p(x) \approx \dfrac{k}{Nv}$
- Volume of a sphere with radius ***r*** in ***d*** dimensions is $v(r,d) = \dfrac{2r^d \pi^{d/2}}{\Gamma(d/2+1)}$
- 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

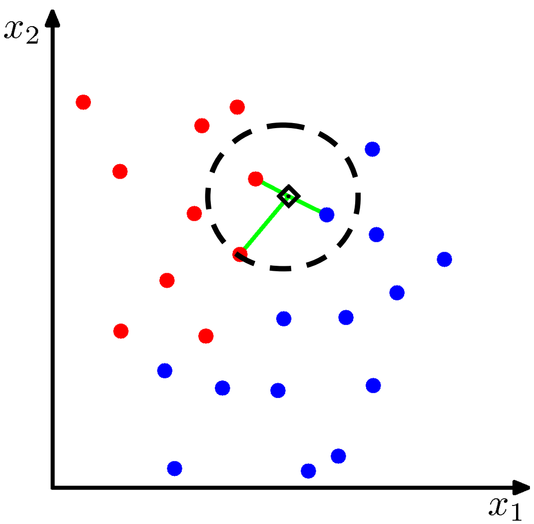
- Use k-nearest neighbor density estimation to find p(x|y)
- Apply Bayes rule for classification: k-nearest neighbor classification

  - Find sphere volume v to capture ***k*** data points for estimate $p(x) = \frac{k}{N\,v}$
  - 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 as fraction of k neighbors in class c

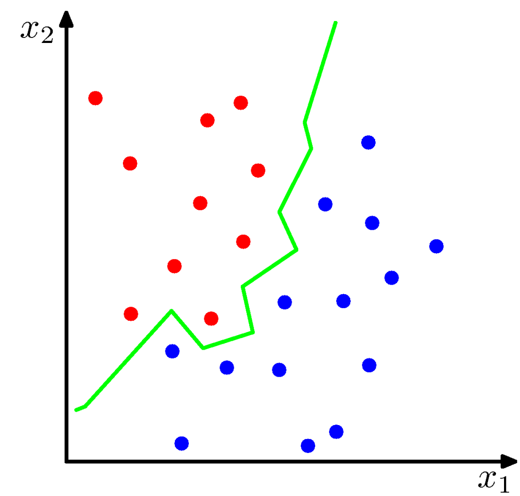
$$p(y=c|x) = \frac{p(y=c)\,p(x|y=c)}{p(x)}$$

$$= \frac{1}{p(x)} \frac{k_c}{Nv}$$

$$= \frac{k_c}{k}$$



(a)



(b)

Summary generative classification methods

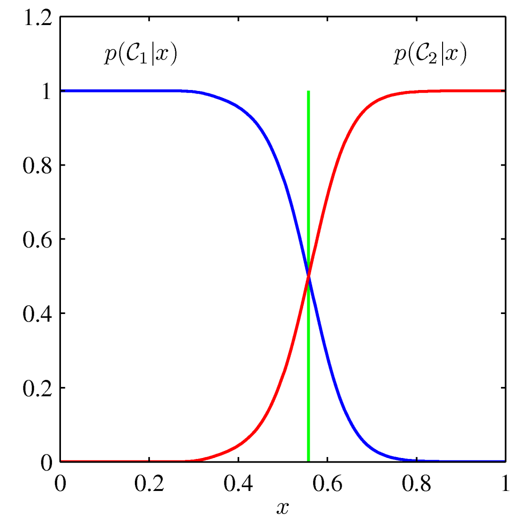
- (Semi-) Parametric models, eg p(x|y) is Gaussian, or mixture of …
  - Pros: no need to store training data, just the class conditional models
  - Cons: may fit the data poorly, and might therefore lead to poor classification result

- 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

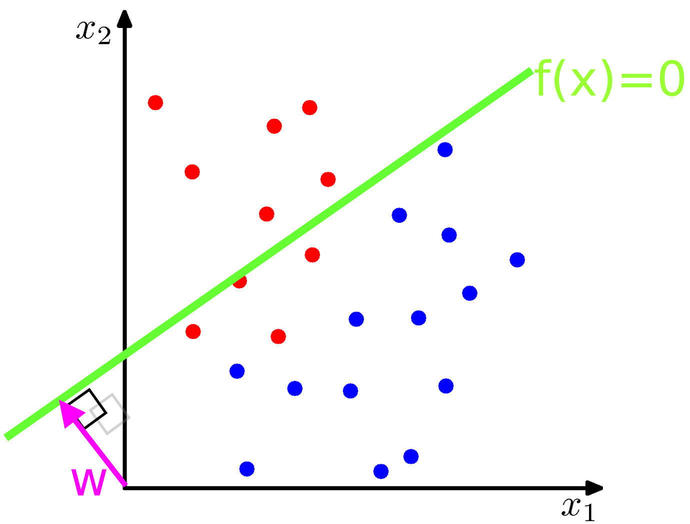
- **Discriminative methods** directly estimate class probability given input: p(y|x)
  - Choose class of decision functions in feature space
  - Estimate function to maximize performance on the training set
  - Classify a new pattern on the basis of this decision rule.



Binary linear classifier

- 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?

Binary linear classifier

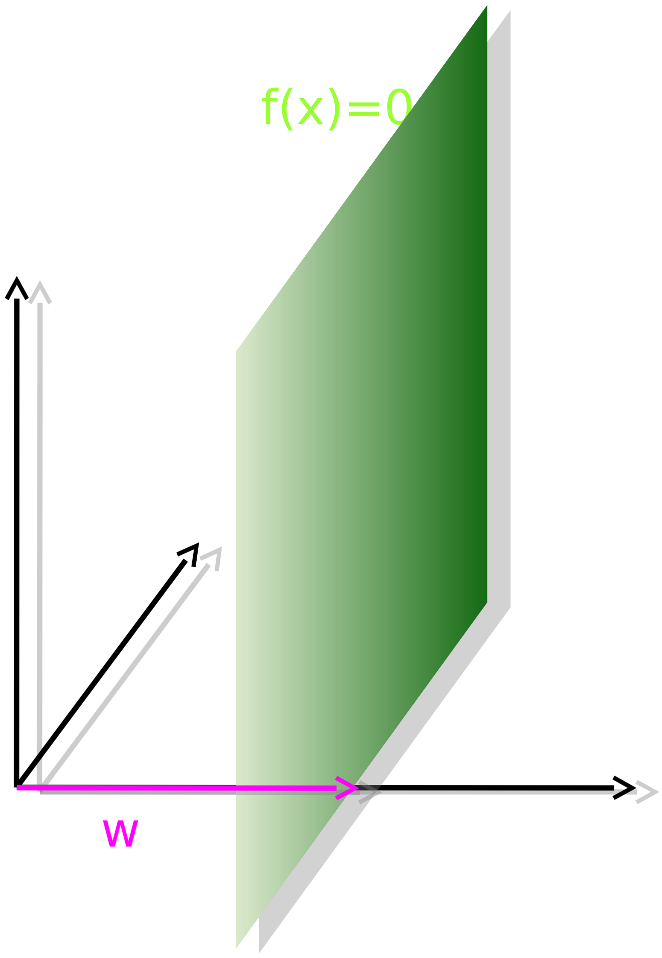
- Decision surface for w=(1,0,0) and b = -1

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

$$b + \sum_{i=1}^{d} w_i x_i = 0$$

$$x_1 - 1 = 0$$

$$x_1 = 1$$

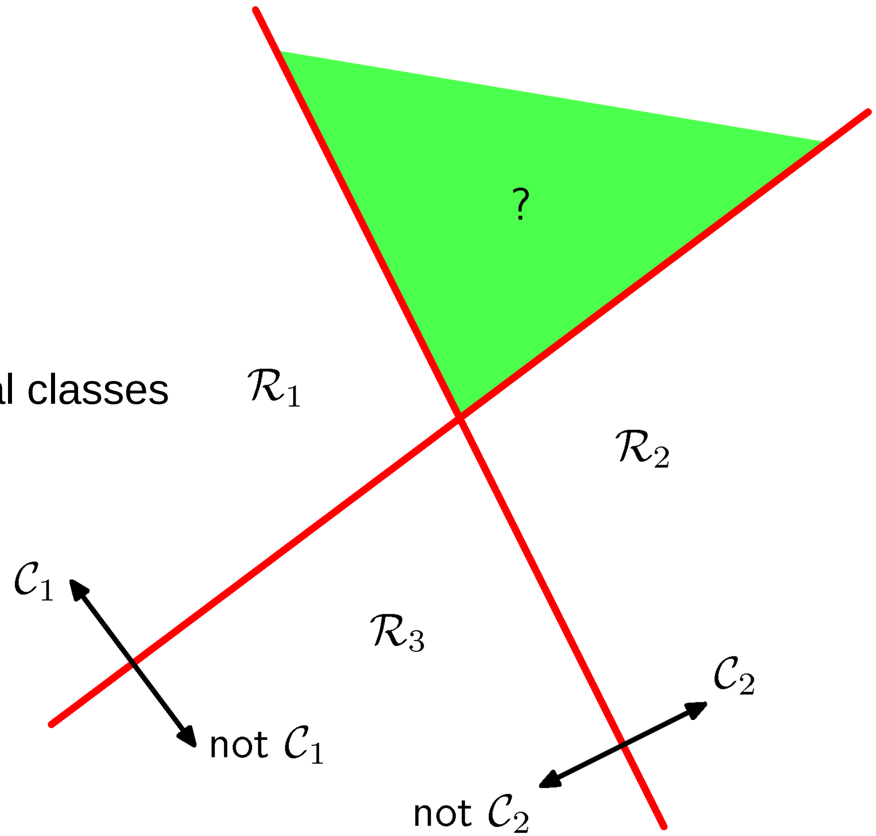


Dealing with more than two classes

- 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

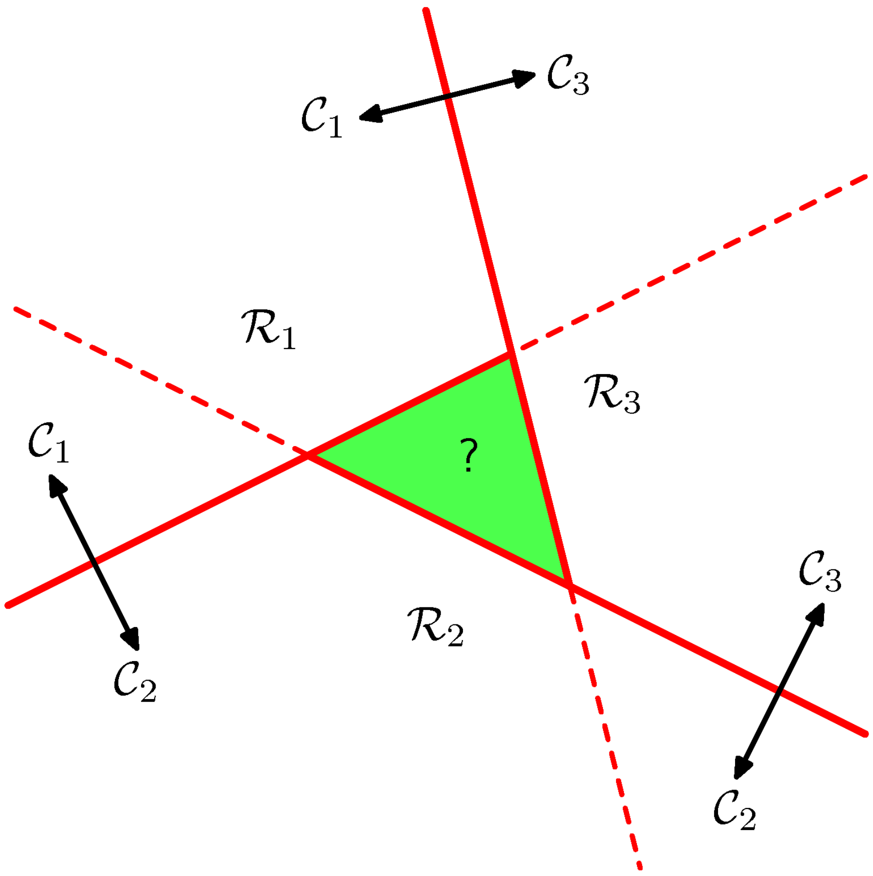


Dealing with more than two classes

- 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



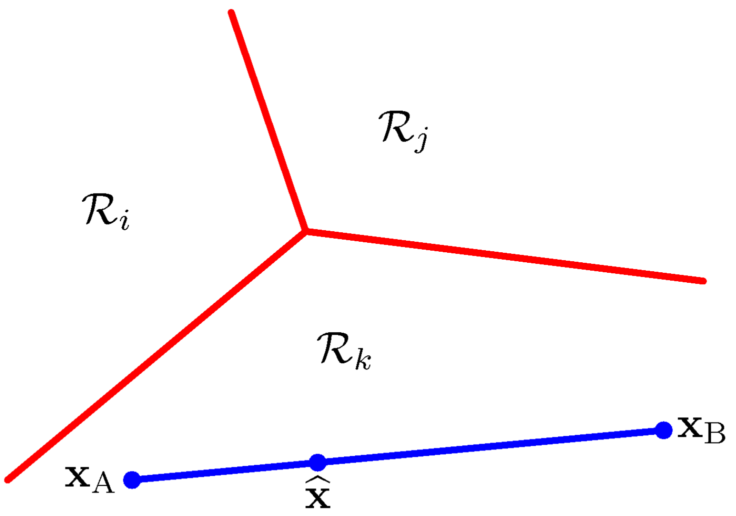
Dealing with more than two classes

- Instead: 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

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

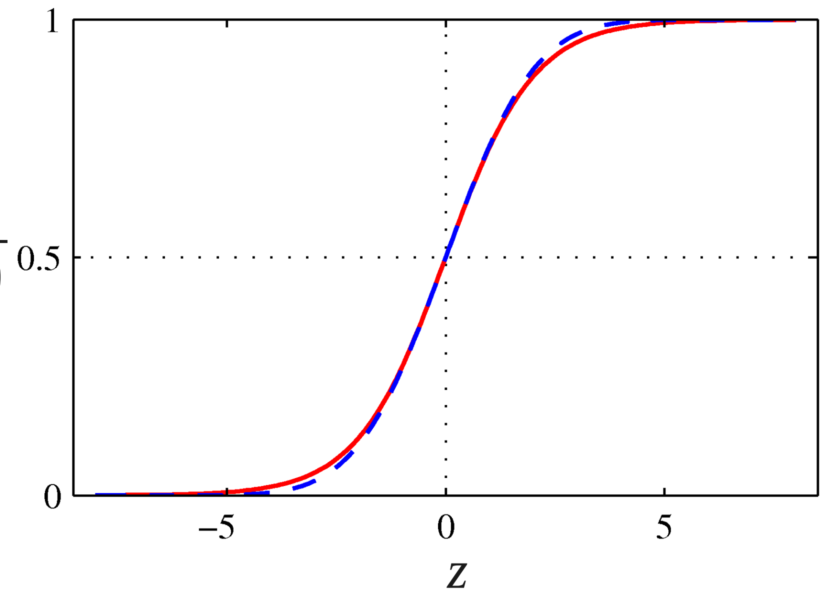
  - For binary classification problem, we have by definition

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

  - Exercise: show that

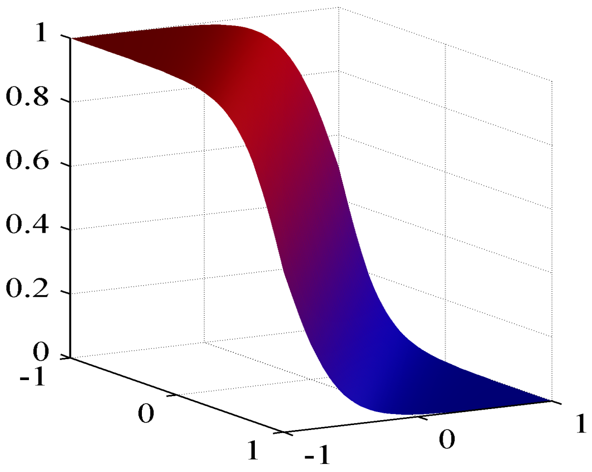
$$p(y=-1|x)=\sigma(-(w^T x+b))$$

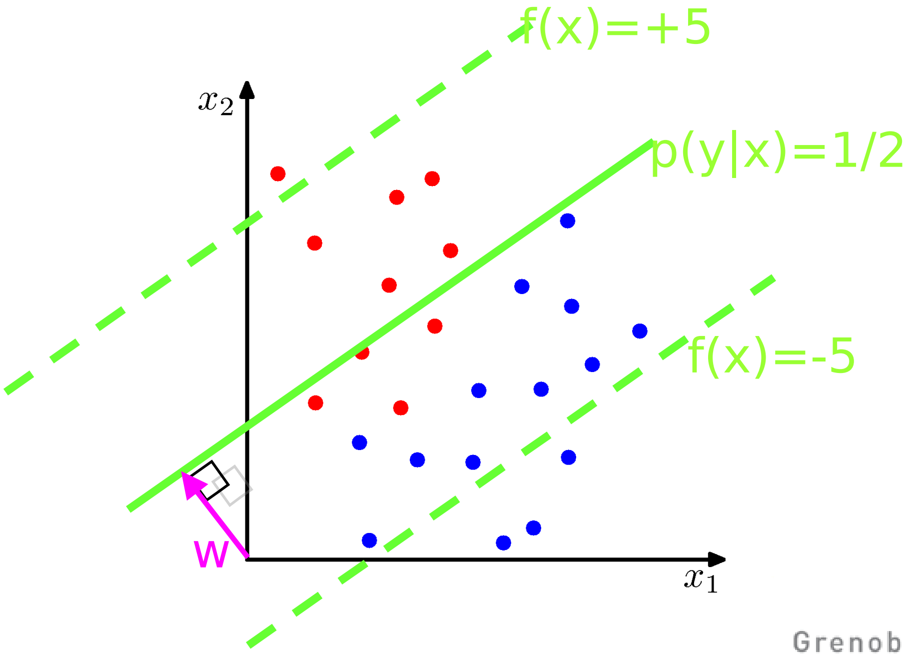
$$\sigma(z)=\frac{1}{1+\exp(-z)}$$



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





informatics mathematics
Grenoble INP Ensimag

Multi-class logistic discriminant

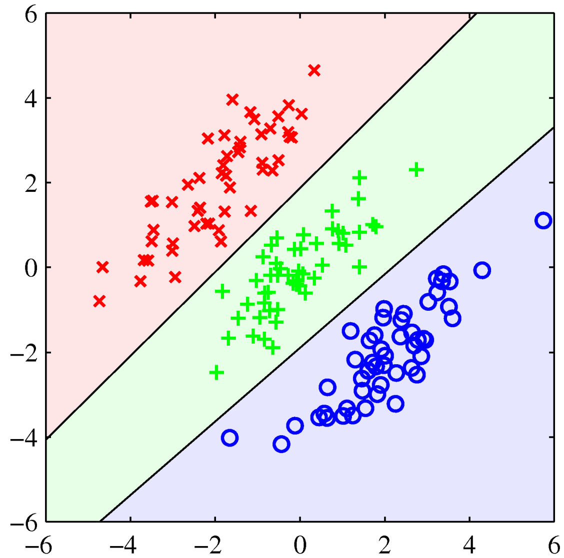
- Map score function of each class to class probabilities with "soft-max" function

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

  - 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



Maximum likelihood parameter estimation

- Maximize the log-likelihood of predicting the correct class label for training data
  - Predictions are made independently, so 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

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

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

Indicator function
1 if y_n=k, else 0

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

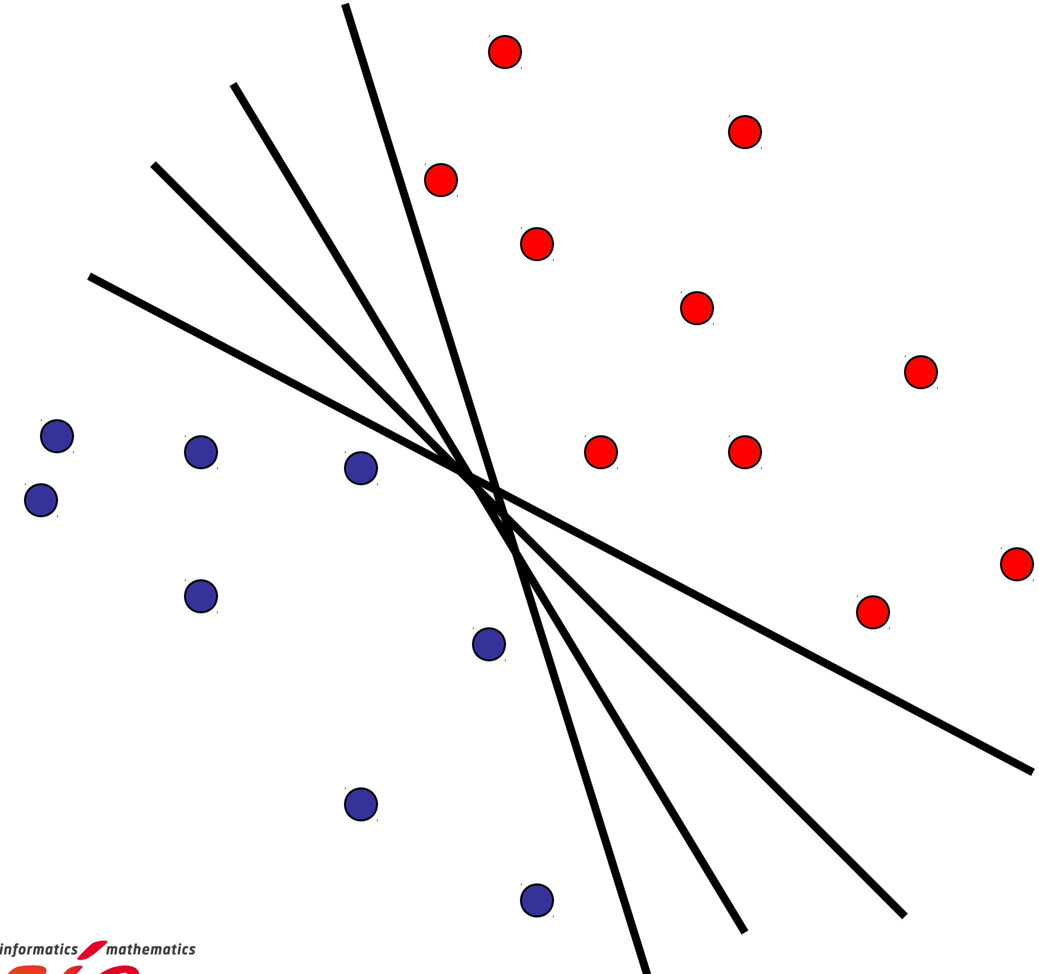
Expected value of each feature, weighting points by p(y|x), should equal empirical expectation.

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

Support Vector Machines

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

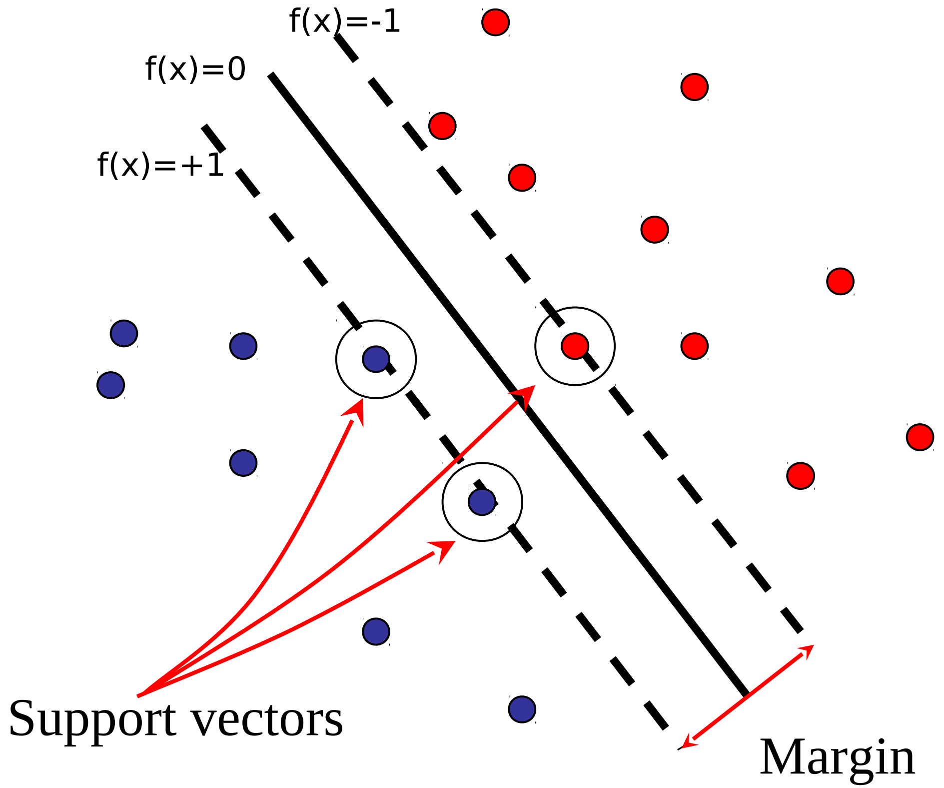
$$y_i = +1 \quad : \quad w^T x + b > 0$$
$$y_i = -1 \quad : \quad w^T x + b < 0$$



Which hyperplane is best?

Support vector machines

- Find maximum margin hyperplane between positive and negative examples
  - Constrain points to be on correct side of boundary $y_i(w^T x + b) \geq 1$
  - Define support vectors as the closest points to the boundary $w^T x + b = y_i$
  - Then it follows that (exercise to show this) margin size is $2/\|w\|$
  - To maximize margin, minimize the norm of w



Finding the maximum margin hyperplane

1. Minimize the norm of w
2. Correctly classify all training data:

$$y_i = +1 \quad : \quad w^T x + b \geq +1$$
$$y_i = -1 \quad : \quad w^T x + b \leq -1$$

Quadratic optimization problem:

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

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

informatics mathematics

Grenoble INP Ensimag

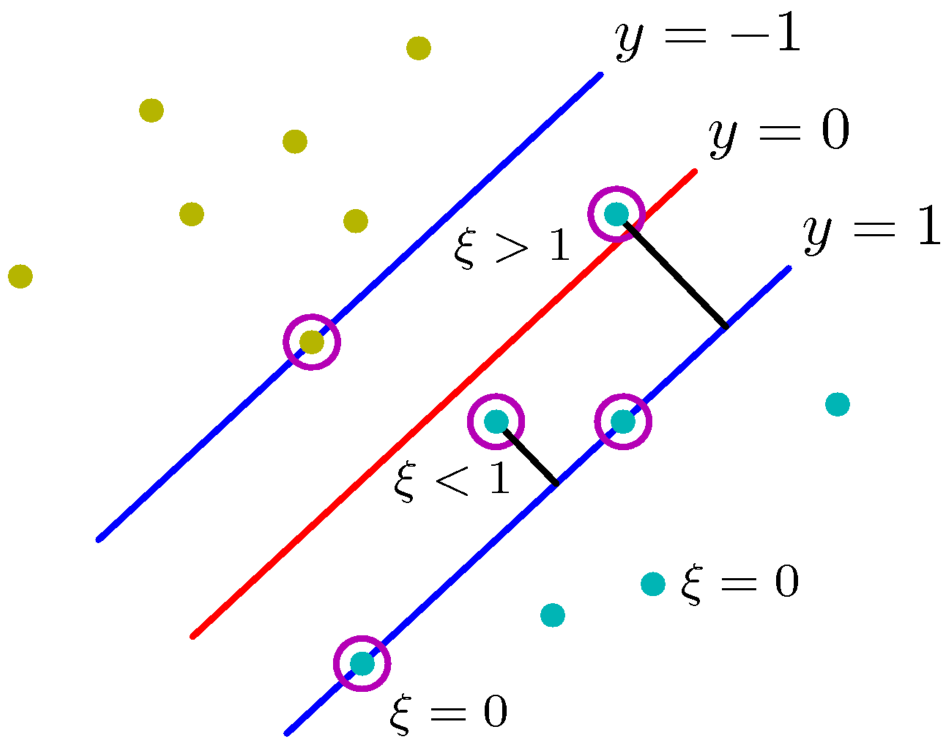
Support vector machines

- **For non-separable classes**: pay a penalty for crossing the margin

$$\xi_i = max(0, 1 - y_i f(x_i))$$

  - If on correct side of the margin: zero
  - Otherwise, amount by which score violates the constraint of correct classification

$$y_i f(x_i) \geq 1$$



Finding the maximum margin hyperplane

- Minimize norm of w, plus penalties:

$$\min_{w,b} \quad \frac{1}{2} w^T w \;+\; C \sum_i \max\left(0, 1 - y_i\left(w^T x + b\right)\right)$$

- Optimization: still a quadratic-programming problem

- C: trades-off between large margin & small penalties
  - Typically set by cross-validation

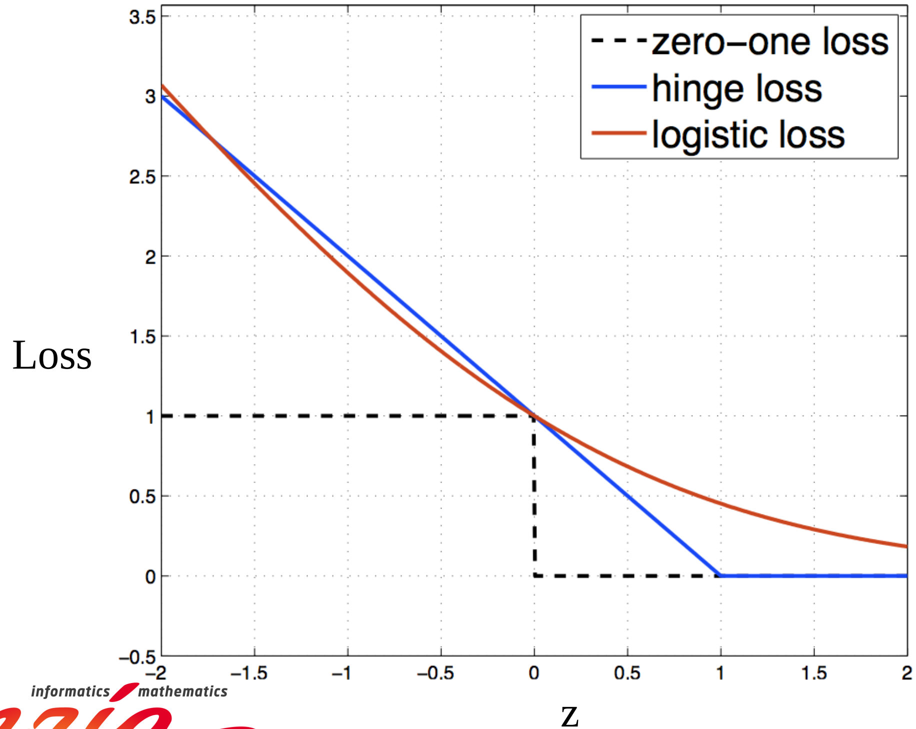
SVM solution properties

- Optimal w is a linear combination of data points $w = \sum_{n=1}^{N} \alpha_n y_n x_n$
- Weights (alpha) are zero for all points on the correct side of the margin
  - Points on the margin also have non-zero weight
- Classification function thus has form $f(x) = w^T x + b = \sum_{n=1}^{N} \alpha_n y_n \boxed{x_n^T x} + b$
  - relies only on inner products between the test point ***x*** and data points with non-zero alpha's
- Solving the optimization problem also requires access to the data only in terms of inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between pairs of training points

Relation SVM and logistic regression

- A classification error occurs when sign of the function does not match the sign of the class label: the zero-one loss $z = y_i f(x_i) \leq 0$

- Consider error minimized when training classifier:
  - Non-separable SVM, hinge loss: $\xi_i = max(0, 1 - y_i f(x_i)) = max(0, 1 - z)$
  - Logistic loss: $-\log p(y_i | x_i) = -\log \sigma(y_i f(x_i)) = \log(1 + \exp(-z))$



- Both hinge & logistic loss are convex bounds on zero-one loss which is non-convex and discontinuous

- Both lead to efficient optimization
  - Hinge-loss is piece-wise linear: quadratic programming
  - Logistic loss is smooth: gradient descent methods

Summary of discriminative linear classification

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

- For both, in the case of binary classification
  - Criterion that is minimized is a convex bound on zero-one loss
  - weight vector **w** is a linear combination of the data points $w = \sum_{n=1}^{N} \alpha_n x_n$

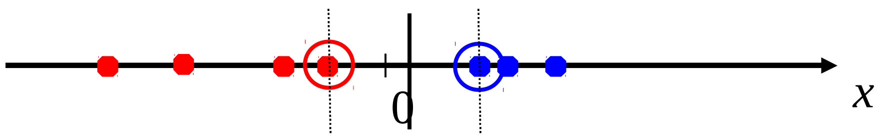
- This means that we only need the inner-products between data points to calculate the linear functions

$$
\begin{aligned}
f(x) &= w^T x + b \\
&= \sum_{n=1}^{N} \alpha_n x_n^T x + b \\
&= \sum_{n=1}^{N} \alpha_n k(x_n, x) + b
\end{aligned}
$$

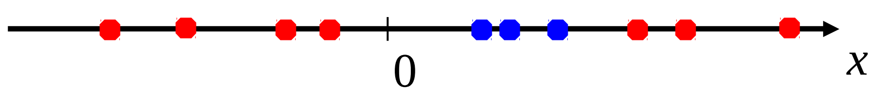
  - The “kernel” function k(,) computes the inner products

Nonlinear Classification

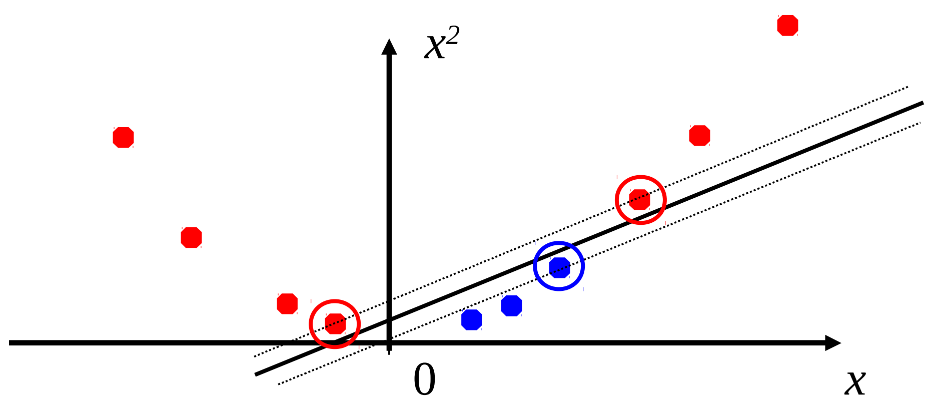
- 1 dimensional data that is linearly separable



- But what if the data is not linearly seperable?



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

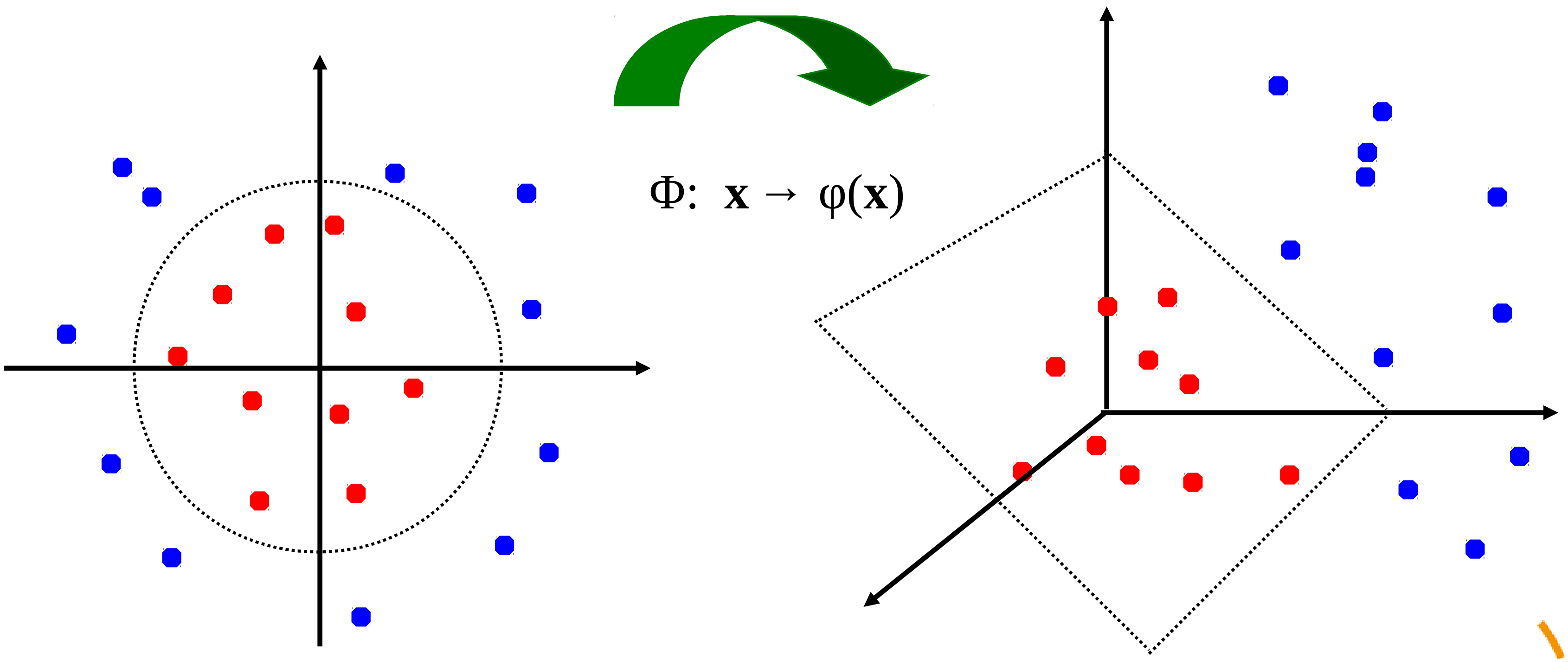


Slide credit: Andrew Moore

Kernels for non-linear classification

- General idea: map the original input space to some higher-dimensional feature space where the training set is separable

- Exercise: find features that could separate the 2d data linearly



Slide credit: Andrew Moore

Nonlinear classification with kernels

- *The kernel trick*: instead of explicitly computing the feature transformation $\boldsymbol{\varphi}(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i) \cdot \boldsymbol{\varphi}(\mathbf{x}_j)$$

- Conversely, if a kernel satisfies Mercer’s condition then it computes an inner product in some feature space, possibly with large or infinite # of dimensions
  - *Mercer's Condition: The square N x N matrix with kernel evaluations for any arbitrary N data points should always be a positive definite matrix.*

- This gives a **nonlinear decision boundary** in the original space:

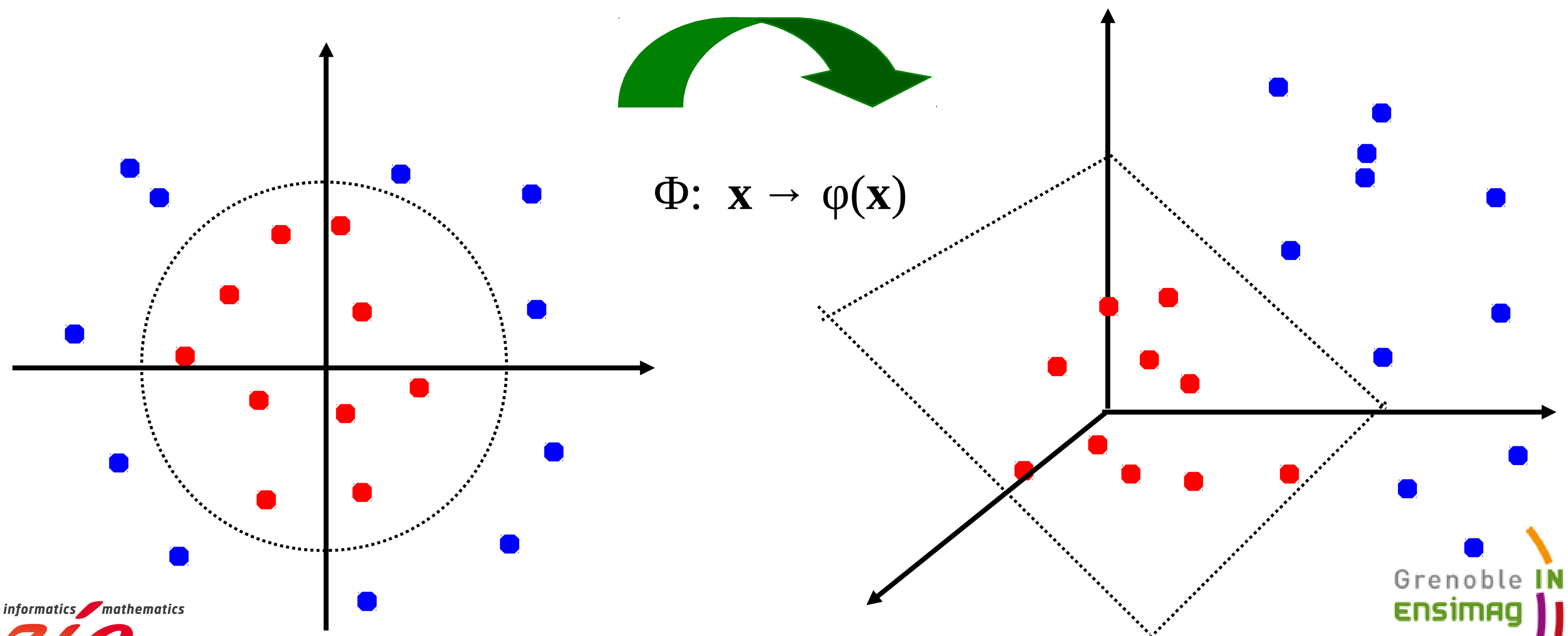
$$\begin{aligned} f(x) &= b + w^T \varphi(x) \\ &= b + \sum_i \alpha_i \varphi(x_i)^T \varphi(x) \\ &= b + \sum_i \alpha_i k(x_i, x) \end{aligned}$$

Kernels for non-linear classification

- What is the kernel function that corresponds to this feature mapping ?

$$\varphi(x) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \end{pmatrix}$$

$$\begin{aligned} k(x, y) &= \varphi(x)^T \varphi(y) = ? \\ &= x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 \\ &= (x_1 y_1 + x_2 y_2)^2 \\ &= (x^T y)^2 \end{aligned}$$



Kernels for non-linear classification

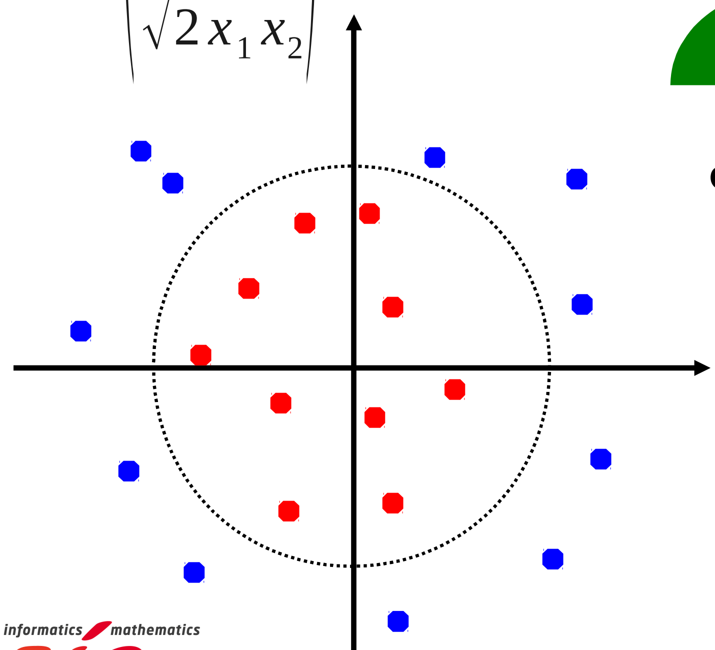
- Suppose we also want to keep the original features to be able to still implement linear functions

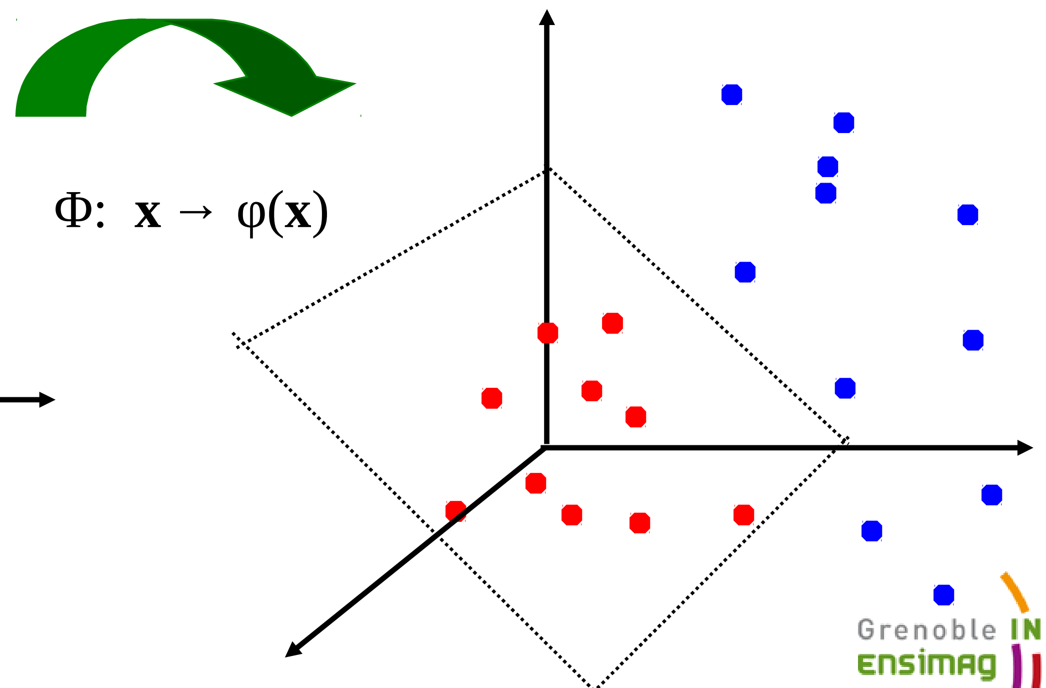
$$k(x,y)=\varphi(x)^T\varphi(y)=?$$

$$=1+2x^Ty+\left(x^Ty\right)^2$$

$$=\left(x^Ty+1\right)^2$$

$$\varphi(x)=\begin{pmatrix}1\\ \sqrt{2}\,x_1\\ \sqrt{2}\,x_2\\ x_1^2\\ x_2^2\\ \sqrt{2}\,x_1x_2\end{pmatrix}$$





Kernels for non-linear classification

- What happens if we use the same kernel for higher dimensional data
  - Which feature vector $\varphi(x)$ corresponds to it ?

$$k(x,y)=\left(x^T y+1\right)^2=1+2x^T y+\left(x^T y\right)^2$$

  - First term, encodes an additional 1 in each feature vector
  - Second term, encodes scaling of the original features by sqrt(2)
  - Let's consider the third term $\left(x^T y\right)^2=\left(x_1 y_1+\ldots+x_D y_D\right)^2$

$$=\sum_{d=1}^{D}\left(x_d y_d\right)^2+2\sum_{d=1}^{D}\sum_{i=d+1}^{D}\left(x_d y_d\right)\left(x_i y_i\right)$$

$$=\sum_{d=1}^{D} x_d^2 y_d^2+2\sum_{d=1}^{D}\sum_{i=d+1}^{D}\left(x_d x_i\right)\left(y_d y_i\right)$$

  - In total we have 1 + 2D + D(D-1)/2 features !
  - But the kernel is computed as efficiently as dot-product in original space

$$\varphi(x)=\left(1,\underbrace{\sqrt{2}x_1,\sqrt{2}x_2,\ldots,\sqrt{2}x_D}_{\text{Original features}},\underbrace{x_1^2,x_2^2,\ldots,x_D^2}_{\text{Squares}},\underbrace{\sqrt{2}x_1x_2,\ldots,\sqrt{2}x_1x_D,\ldots,\sqrt{2}x_{D-1}x_D}_{\text{Products of two distinct elements}}\right)^T$$

Popular kernels for bags of features

- Hellinger kernel:

$$k(h_1, h_2) = \sum_d \sqrt{h_1(i)} \times \sqrt{h_2(i)}$$

- Histogram intersection kernel:

$$k(h_1, h_2) = \sum_d min(h_1(d), h_2(d))$$

  - Exercise: find the feature transformation ?

- Generalized Gaussian kernel:

$$k(h_1, h_2) = \exp\left(-\frac{1}{A} d(h_1(i), h_2(i))\right)$$

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

See also:
J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid,
Local features and kernels for classification of texture and object categories: a comprehensive study. Int. Journal of Computer Vision, 2007

Summary linear classification & kernels

- Linear classifiers learned by minimizing convex cost functions
  - Logistic discriminant: smooth objective, minimized using gradient descend
  - Support vector machines: piecewise linear objective, quadratic programming
  - Both require only computing inner product between data points

- Non-linear classification can be done with linear classifiers over new features that are non-linear functions of the original features
  - Kernel functions efficiently compute inner products in (very) high-dimensional spaces, can even be infinite dimensional in some cases.

- Using kernel functions non-linear classification has drawbacks
  - Requires storing the support vectors, may cost lots of memory in practice
  - Computing kernel between new data point and support vectors may be computationally expensive (at least more expensive than linear classifier)

- Kernel functions also work for other linear data analysis techniques
  - Principle component analysis, k-means clustering, ….

Reading material

- A good book that covers all machine learning aspects of the course is
  - Pattern recognition & machine learning
    Chris Bishop, Springer, 2006

- For clustering with k-means & mixture of Gaussians read
  - Section 2.3.9
  - Chapter 9, except 9.3.4
  - Optionally, Section 1.6 on information theory

- For classification read
  - Section 2.5, except 2.5.1
  - Section 4.1.1 & 4.1.2
  - Section 4.2.1 & 4.2.2
  - Section 4.3.2 & 4.3.4
  - Section 6.2
  - Section 7.1 start + 7.1.1 & 7.1.2