Support Vector Machines & Fisher Vector image representation

Machine Learning and Category Representation 2012-2013

Jakob Verbeek, December 14, 2012

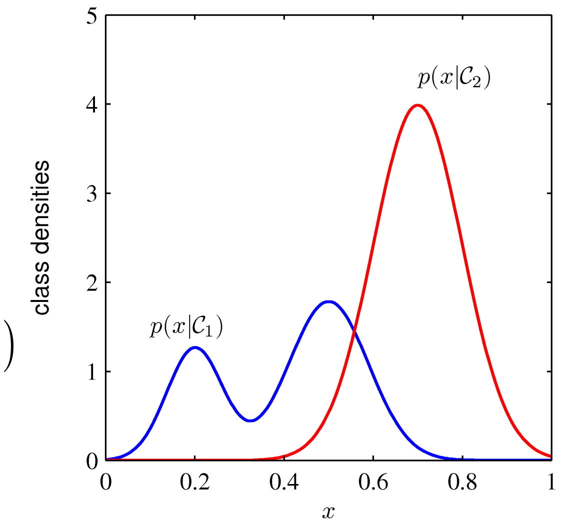
Course website:

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

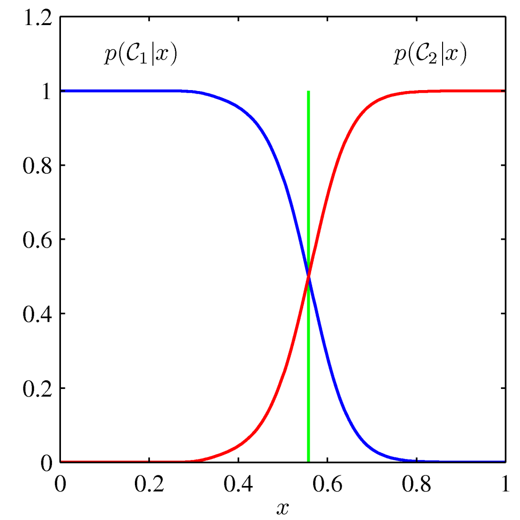
Discriminative vs generative methods

- 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

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



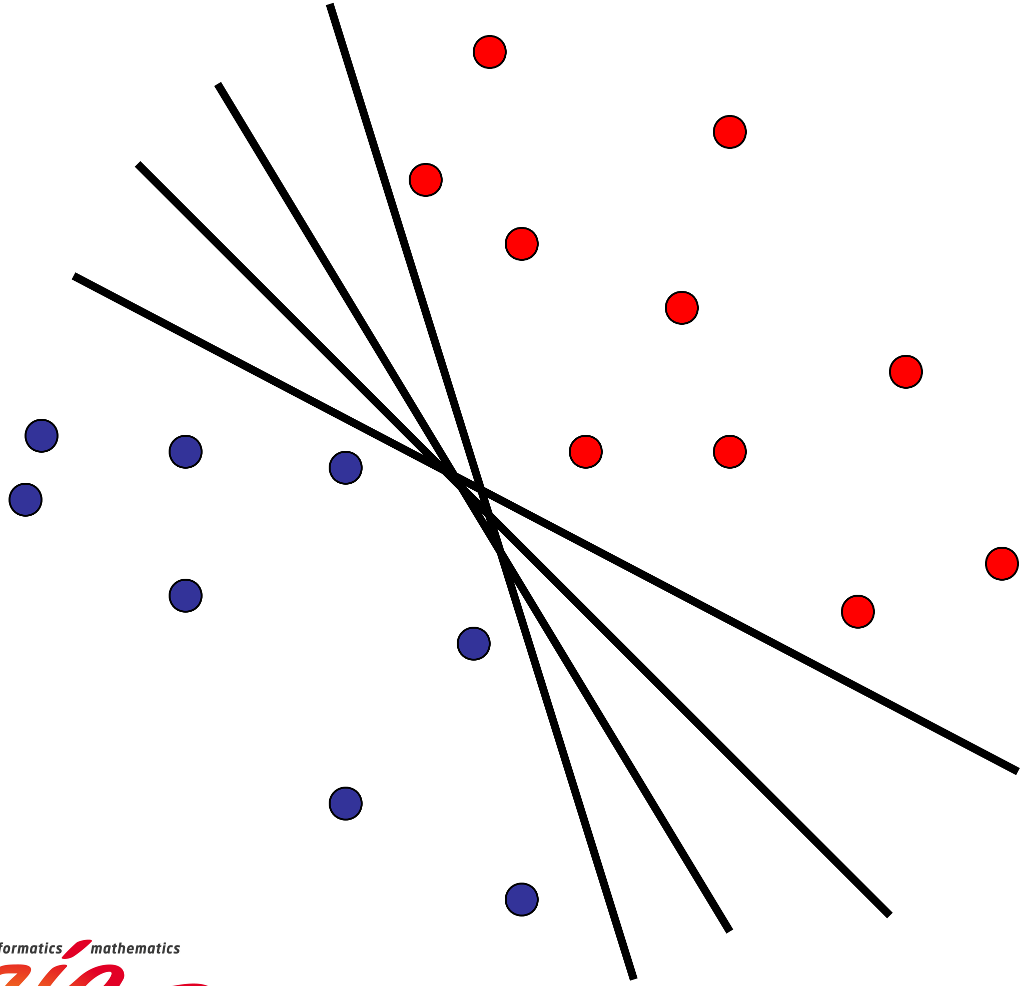
- **Discriminative (probabilistic) methods**
  - Directly estimate class probability given input: p(y|x)
  - Some methods do not have probabilistic interpretation,
    - eg. they fit a function f(x), and assign to class 1 if f(x)>0, and to class 2 if f(x)<0



Support Vector Machines

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

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

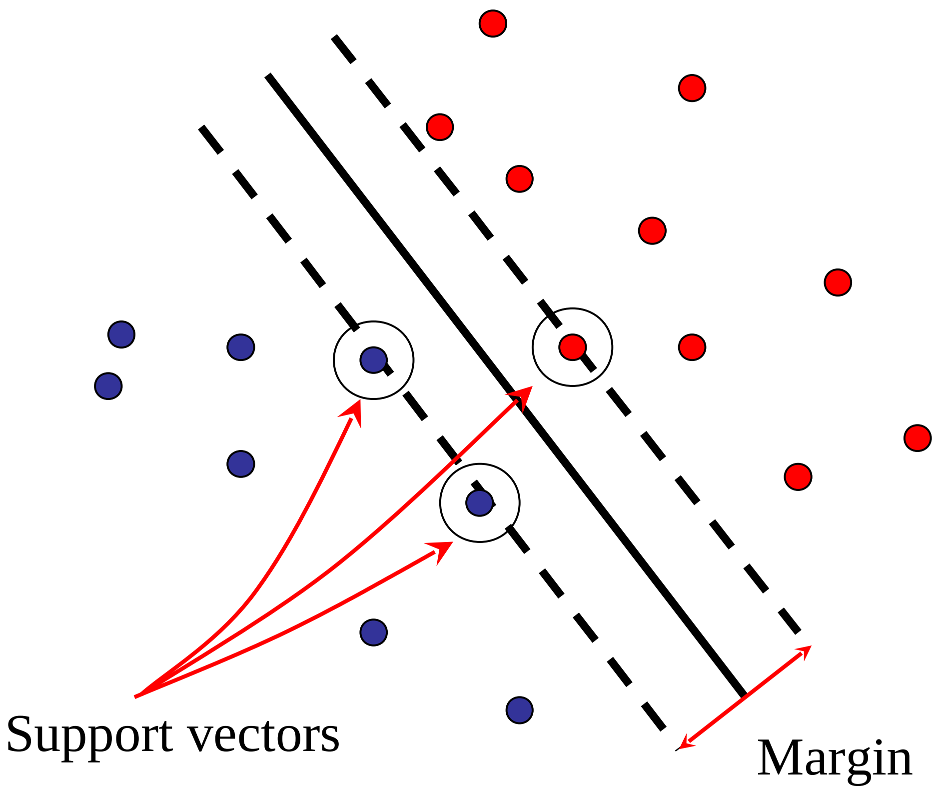


Which hyperplane
is best?

Support vector machines

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

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



For support vectors $\quad w^T x + b = y_i$

Distance between point and hyperplane: $\quad \dfrac{|w^T x + b|}{\|w\|}$

Therefore, the margin is $\quad \dfrac{2}{\|w\|}$

Exercise: show this

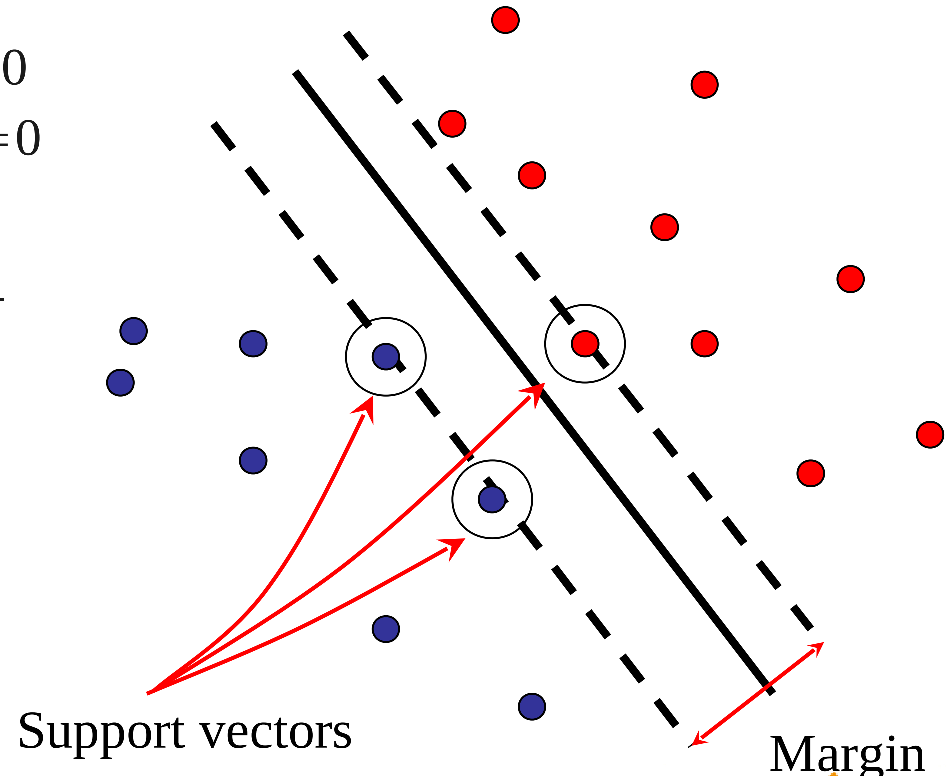
Support vector machines

- Let's consider a support vector x from the positive class $f(x) = w^T x + b = 1$
- Let z be its projection on the decision plane
  - Since w is normal vector to the decision plane, we have $z = x - \alpha w$
  - and since z is on the decision plane $f(z) = w^T(x - \alpha w) + b = 0$
- Solve for alpha

$$w^T(x - \alpha w) + b = 0$$
$$w^T x + b - \alpha w^T w = 0$$
$$1 - \alpha w^T w = 0$$
$$\alpha = \frac{1}{w^T w} = \frac{1}{\|w\|^2}$$

- Margin is twice distance from x to z

$$\|x - z\| = \|x - (x - \alpha w)\|$$
$$\|\alpha w\| = \alpha \|w\|$$
$$\frac{\|w\|}{\|w\|^2} = \frac{1}{\|w\|}$$



Finding the maximum margin hyperplane

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

$$\begin{array}{lcl} y_i = +1 & : & w^T x + b \geq +1 \\ y_i = -1 & : & w^T x + b \leq -1 \end{array}$$

Quadratic optimization problem:

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

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

Finding the maximum margin hyperplane

- Solution has properties
  - w linear combination of data points $$w=\sum_{n=1}^{N}\alpha_n y_n x_n$$
  - For support vectors $$y_n=w^T x_n+b$$ $$b=y_n-w^T x_n$$

Learned weights
Non-zero only for
Support vectors

- Classification function thus has form

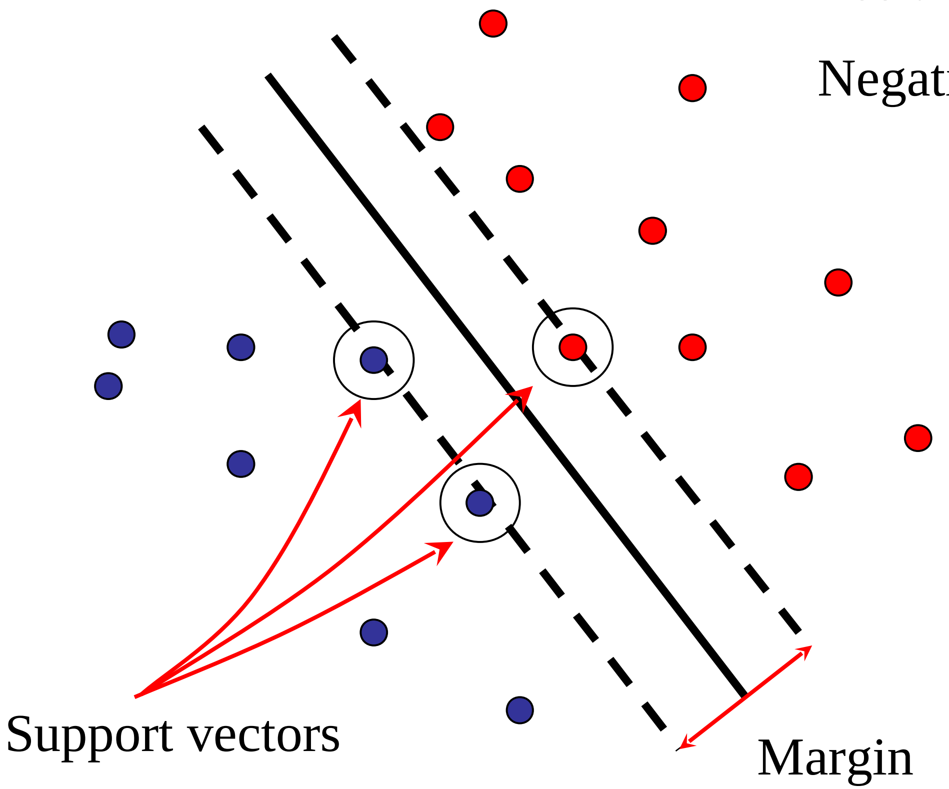
$$f(x)=w^T x+b=\sum_{n=1}^{N}\alpha_n y_n x_n^T x+b$$

  - relies only on inner products between the test point $\boldsymbol{x}$ and data points with non-zero alpha's

- Solving the optimization problem also requires evaluation of the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points

Support vector machines

- **For separable classes**: Find hyperplane that maximizes the *margin* between the positive and negative examples



Positive samples $(y_i = +1)$: $w^T x_i + b \geq +1$

Negative samples $(y_i = -1)$: $w^T x_i + b \leq -1$

For support vectors, $w^T x_i + b = \pm 1$

Maximize the margin, thus minimize $w^T w$

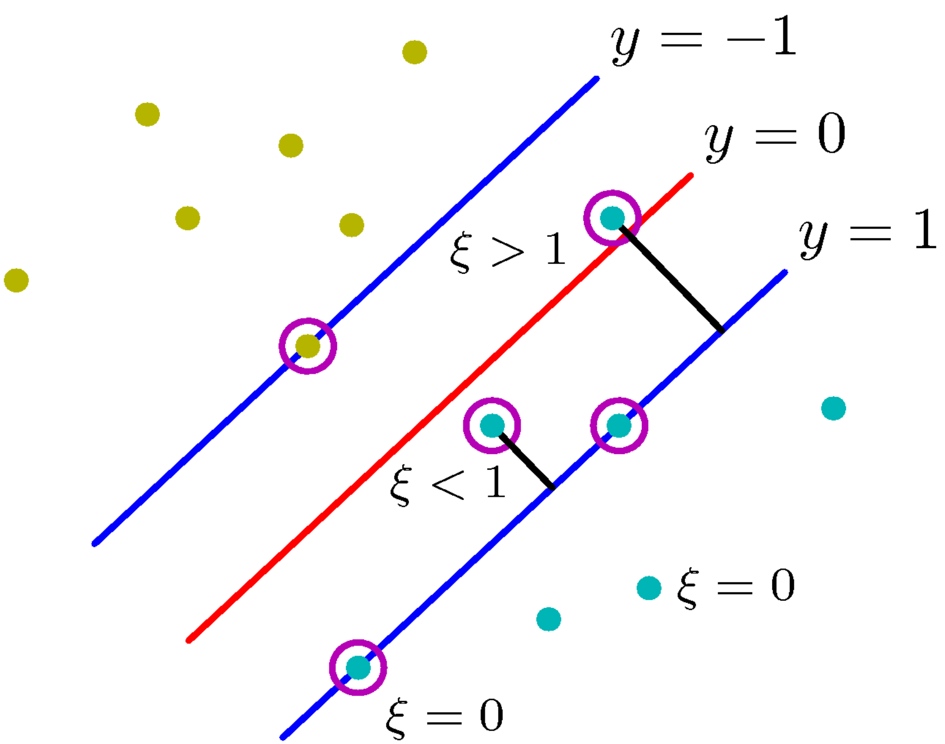
Support vector machines

- **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: $min_{w,\xi_i} \quad \frac{1}{2} w^T w + C \sum_i \xi_i$

- Constraints to correctly classify training data, up to penalties :

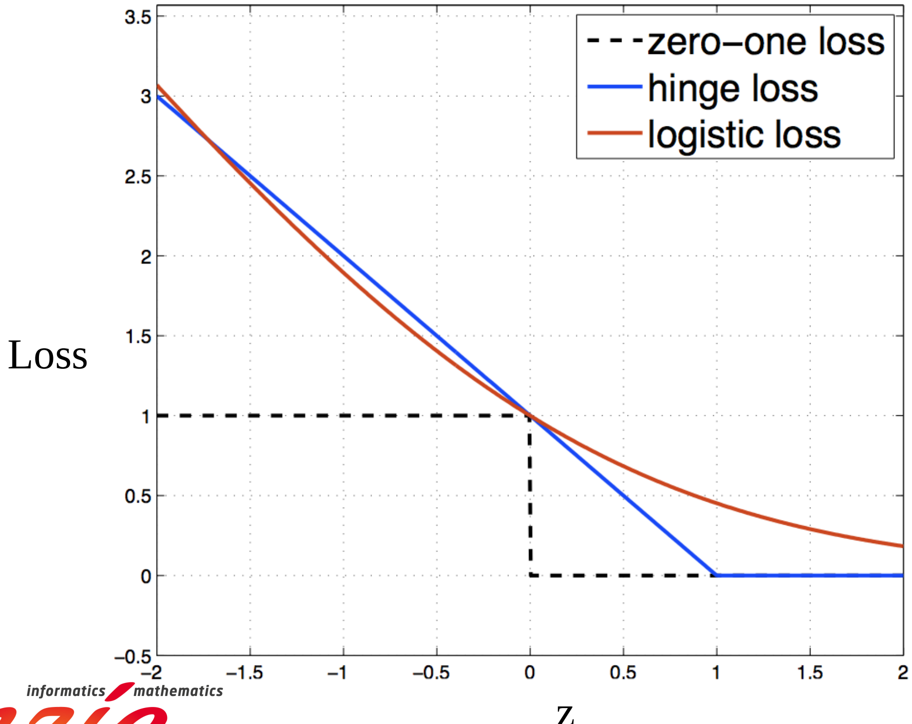
$$y_i f(x_i) = y_i (w^T x_i + b) \geq 1 - \xi_i$$

- Optimization: still a quadratic-programming problem
  - ξ_i : “slack variables”, loosening the margin constraint
  - C: trade-off between large margin & small penalties
    - Typically set by cross-validation, i.e. learn with part of data set using various C values, evaluate classification on held-out data. Repeat for several train-validation splits to pick the optimal C value.

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 recently developed)

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

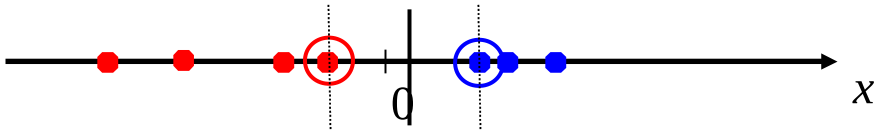
- This means that we only need the inner-products between data points to calculate the linear functions

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

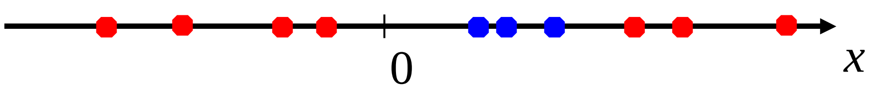
  - The “kernel” function k(,) computes the inner products

Nonlinear Classification

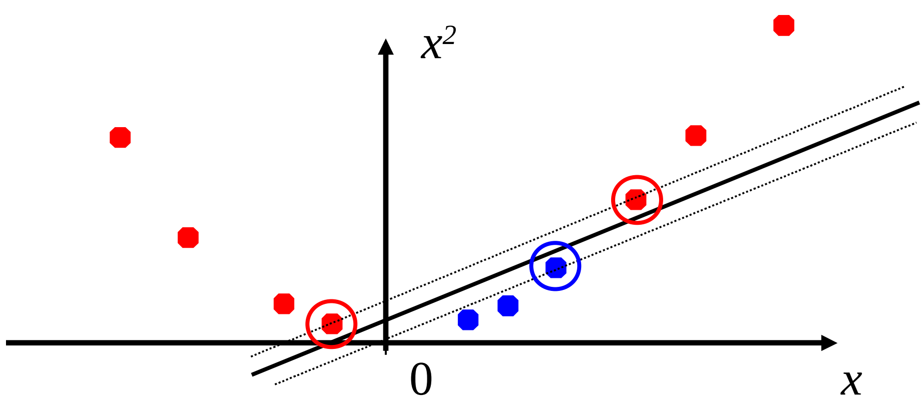
- 1 dimensional data that is linearly separable



- But what if the data is not linearly seperable?



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

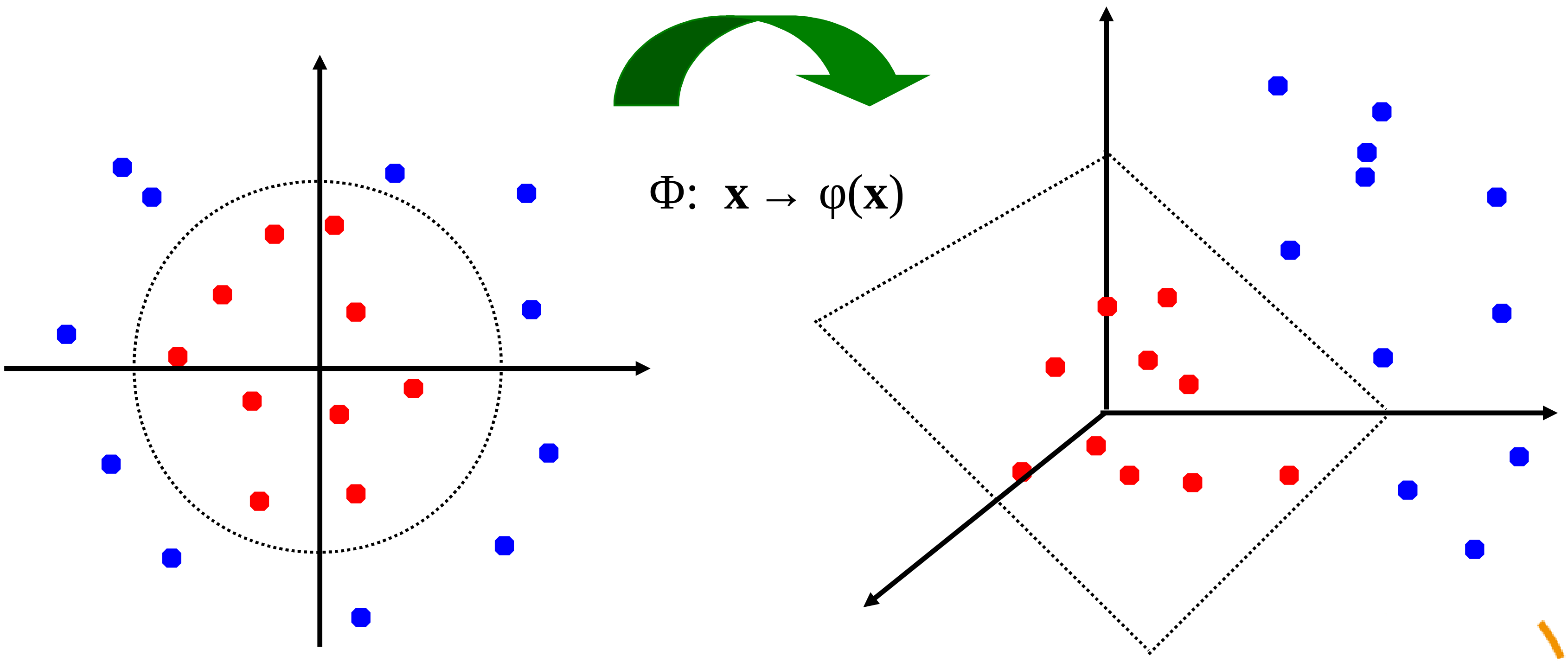


Slide credit: Andrew Moore

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 this data linearly



Slide credit: Andrew Moore

Nonlinear SVMs

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

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

- If a kernel satisfies Mercer's condition then it computes an inner product in some feature space (possibly infinite nr 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:

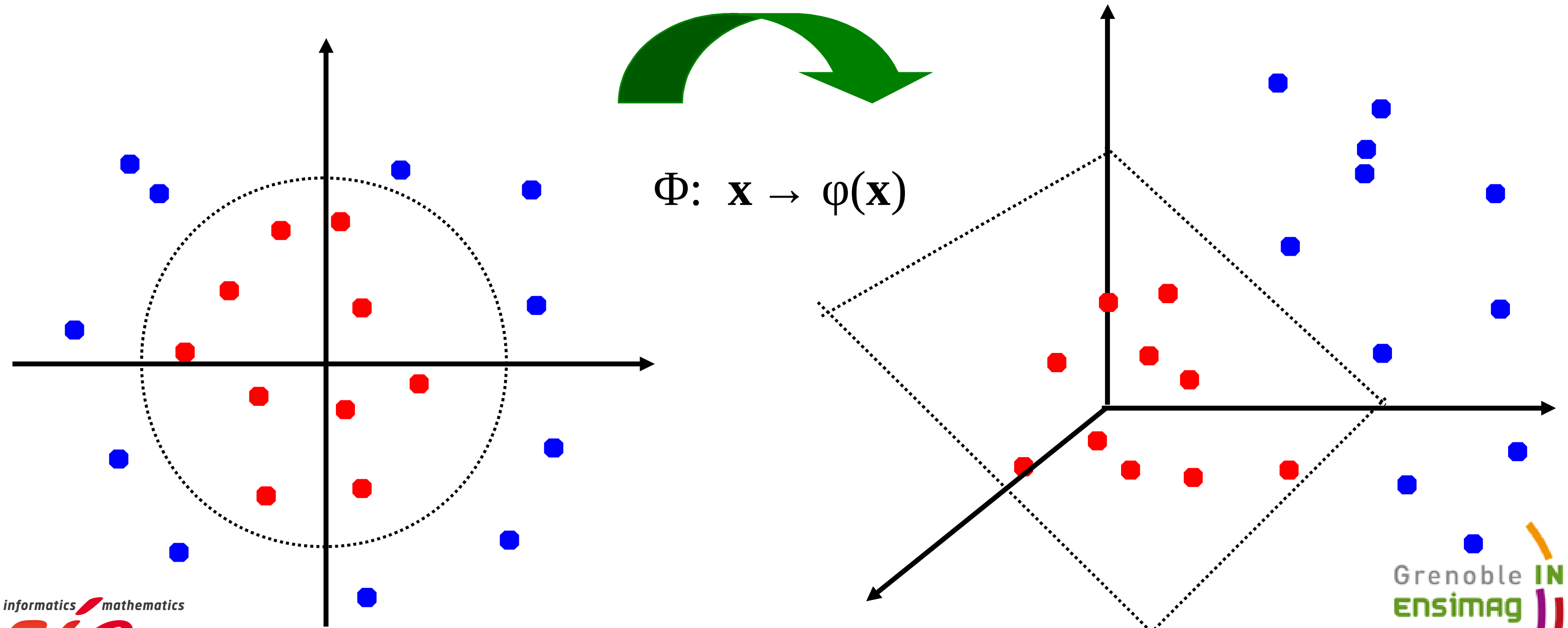
$$\begin{aligned} f(x) &= b + w^T \phi(x) \\ &= b + \sum_i \alpha_i \phi(x_i)^T \phi(x) \\ &= b + \sum_i \alpha_i k(x_i, x) \end{aligned}$$

Kernels for non-linear classification

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

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

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

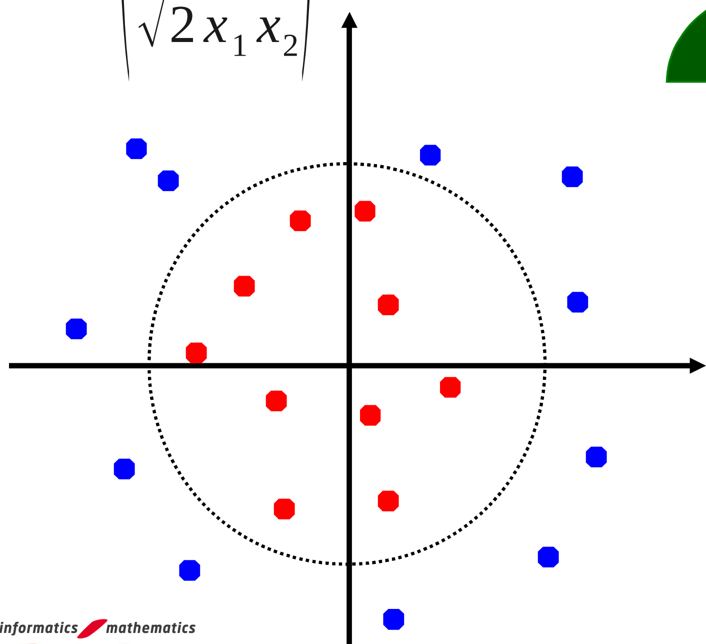


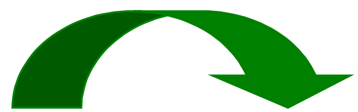
Kernels for non-linear classification

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

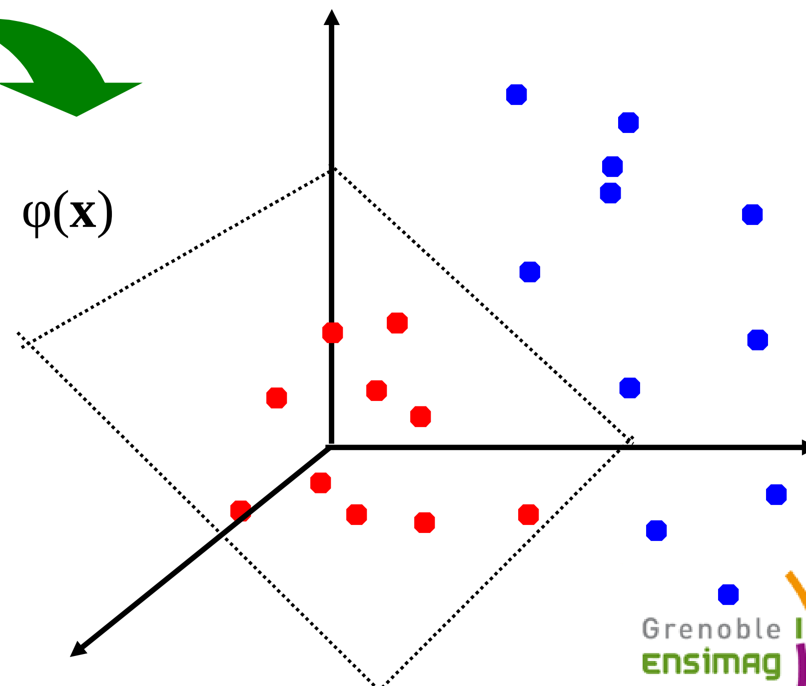
$$k(x,y)=\phi(x)^T\phi(y)=?$$
$$=1+2\mathrm{x}^T y+\left(x^T y\right)^2$$
$$=\left(x^T y+1\right)^2$$

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





$\Phi\colon\ \mathbf{x} \rightarrow \varphi(\mathbf{x})$



Kernels for non-linear classification

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

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

  - 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 inner product did not get any harder to compute

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

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

informatics mathematics
Grenoble INP Ensimag

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, ….

Fisher vector image representation

- An alternative to bag-of-words image representation introduced in

  Fisher kernels on visual vocabularies for image categorization

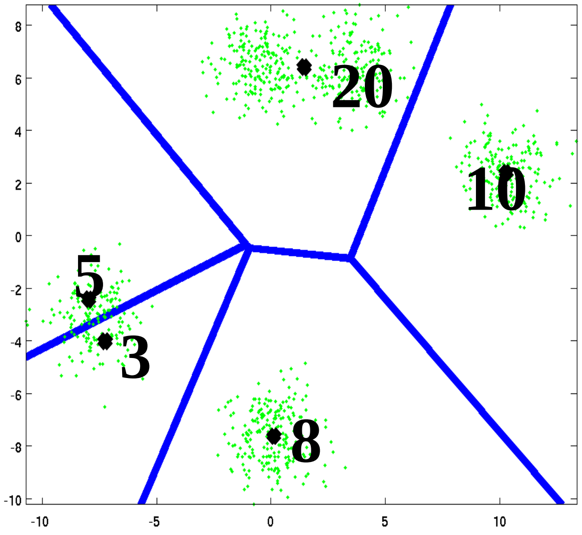
  F. Perronnin and C. Dance, CVPR 2007.

- FV in comparison to the BoW representation
  - Both FV and BoW are based on a visual vocabulary, with assignment of patches to visual words
  - FV based on Mixture of Gaussian clustering of patches, BoW based on k-means clustering
  - FV Extracts a larger image signature than the BoW representation for a given number of visual words
  - Leads to good classification results using linear classifiers, where BoW representations require non-linear classifiers.

Fisher vector representation: Motivation 1

- Suppose we use a bag-of-words image representation
  - Visual vocabulary trained offline

- Feature vector quantization is computationally expensive in practice
- To extract visual word histogram for a new image
  - Compute distance of each local descriptor to each k-means center
  - run-time O(NKD) : linear in
    - N: nr. of feature vectors ~ 10^4 per image
    - K: nr. of clusters ~ 10^3 for recognition
    - D: nr. of dimensions ~ 10^2 (SIFT)



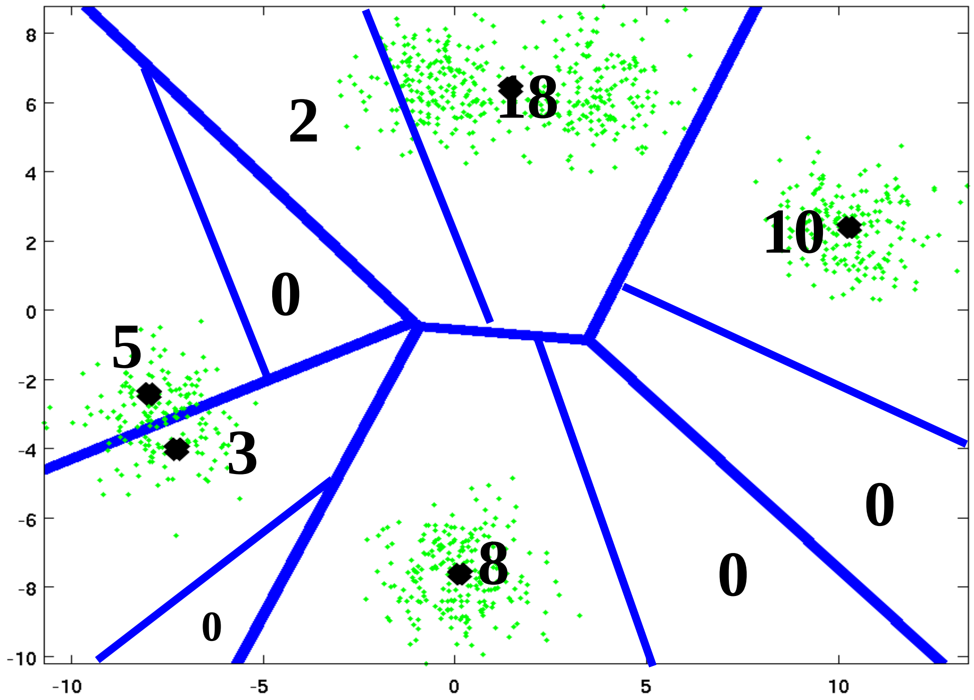
- So in total in the order of 10^9 multiplications per image to obtain a histogram of size 1000

- Can this be done more efficiently ?!
  - Yes, extract more than just a visual word histogram !

Fisher vector representation: Motivation 2

- Suppose we want to refine a given visual vocabulary

- Bag-of-word histogram stores # patches assigned to each word
  - Need more words to refine the representation
  - But this directly increases the computational cost
  - And leads to many empty bins: redundancy



Fisher vector representation in a nutshell

- Instead, the Fisher Vector also records the mean and variance of the points per dimension in each cell
  - More information for same # visual words
  - Does not increase computational time significantly
  - Leads to high-dimensional feature vectors
- Even when the counts are the same the position and variance of the points in the cell can vary

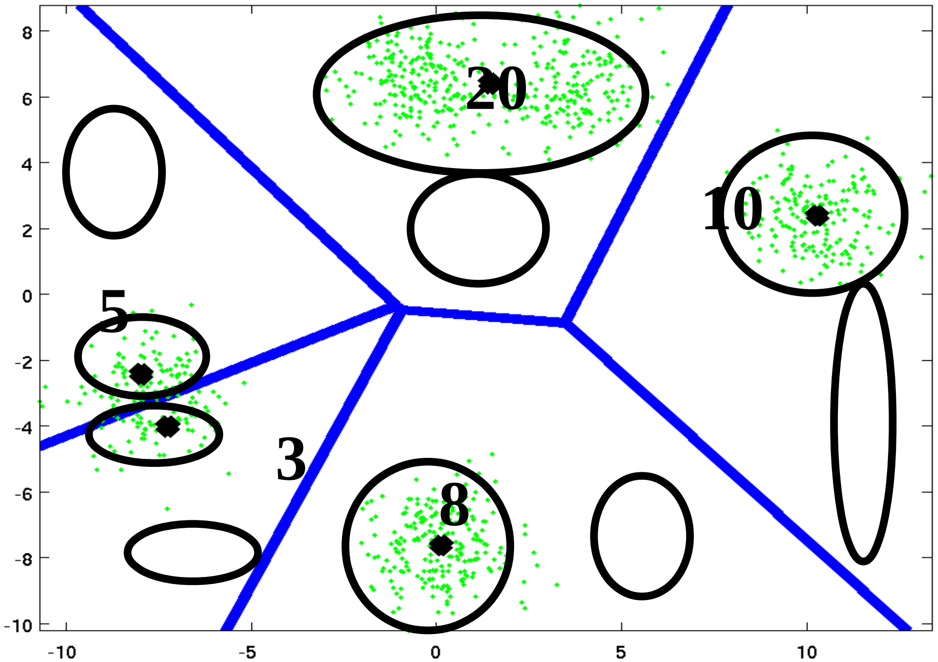


Image representation using Fisher kernels

- General idea of Fischer vector representation
  - Fit probabilistic model to data $p(X;\Theta)$
  - Represent data with derivative of data log-likelihood
    "How does the data want that the model changes?"

$$G(X,\Theta)=\frac{\partial \log p(x;\Theta)}{\partial \Theta}$$

Jaakkola & Haussler. "Exploiting generative models in discriminative classifiers", in Advances in Neural Information Processing Systems 11, 1999.

- Mixture of Gaussians to model the local (SIFT) descriptors $X=\{x_n\}_{n=1}^{N}$

$$L(X,\Theta)=\sum_n \log p(x_n)$$
$$p(x_n)=\sum_k \pi_k N(x_n;m_k,C_k)$$

  - Define mixing weights using the soft-max function
    ensures positiveness and sum to one constraint

$$\pi_k=\frac{\exp \alpha_k}{\sum_{k'} \exp \alpha_{k'}}$$

Image representation using Fisher kernels

- Mixture of Gaussians to model the local (SIFT) descriptors

$$L(\Theta) = \sum_n \log p(x_n)$$
$$p(x_n) = \sum_k \pi_k N(x_n; m_k, C_k)$$

  - The parameters of the model are $\Theta = \{\alpha_k, m_k, C_k\}_{k=1}^{K}$
  - where we use diagonal covariance matrices

- Concatenate derivatives to obtain data representation

$$G(X,\Theta) = \left(\frac{\partial L}{\partial \alpha_1}, \dots, \frac{\partial L}{\partial \alpha_K} \;,\; \frac{\partial L}{\partial m_1}, \dots, \frac{\partial L}{\partial m_K} \;,\; \frac{\partial L}{\partial C_1^{-1}}, \dots, \frac{\partial L}{\partial C_K^{-1}} \right)^T$$

Image representation using Fisher kernels

- Data representation

$$G(X,\Theta)=\left(\frac{\partial L}{\partial \alpha_1},\ldots,\frac{\partial L}{\partial \alpha_K} \;,\; \frac{\partial L}{\partial m_1},\ldots,\frac{\partial L}{\partial m_K} \;,\; \frac{\partial L}{\partial C_1^{-1}},\ldots,\frac{\partial L}{\partial C_K^{-1}}\right)^T$$

- In total K(1+2D) dimensional representation, since for each visual word / Gaussian we have

Count (1 dim) : $\frac{\partial L}{\partial \alpha_k}=\sum_n \left(q_{nk}-\pi_k\right)$

More/less patches assigned to visual word than usual?

Mean (D dims) : $\frac{\partial L}{\partial m_k}=C_k^{-1}\sum_n q_{nk}\left(x_n-m_k\right)$

Center of assigned data relative to cluster center

Variance (D dims) : $\frac{\partial L}{\partial C_k^{-1}}=\frac{1}{2}\sum_n q_{nk}\left(C_k-\left(x_n-m_k\right)^2\right)$

Variance of assigned data relative to cluster variance

With the soft-assignments: $q_{nk}=p\left(k|x_n\right)=\frac{\pi_k\, p\left(x_n|k\right)}{p\left(x_n\right)}$

Bag-of-words vs. Fisher vector image representation

- Bag-of-words image representation
  - Off-line: fit k-means clustering to local descriptors
  - Represent image with histogram of visual word counts: K dimensions

- Fischer vector image representation
  - Off-line: fit MoG model to local descriptors
  - Represent image with gradient of log-likelihood: K(2D+1) dimensions

- Computational cost similar:
  - Both compare N descriptors to K visual words (centers / Gaussians)
- Memory usage: higher for fisher vectors
  - Fisher vector is a factor (2D+1) larger, e.g. a factor 257 for SIFTs !
    - For 1000 visual words the FV has 257,000 dimensions
  - However, because we store more information per visual word, we can generally obtain same or better performance with far less visual words

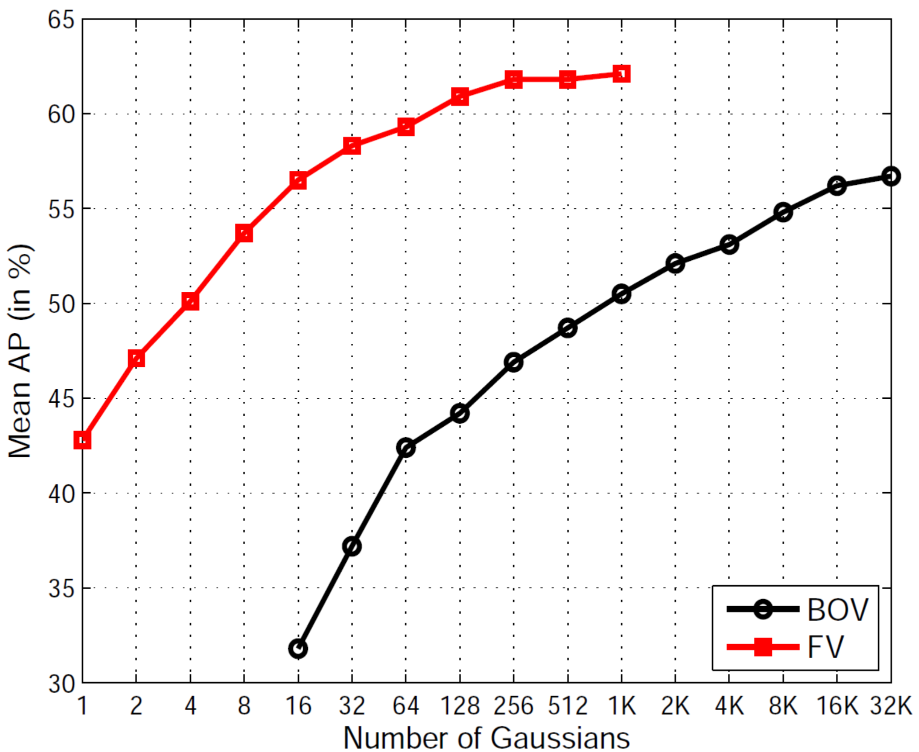
Images from categorization task PASCAL VOC

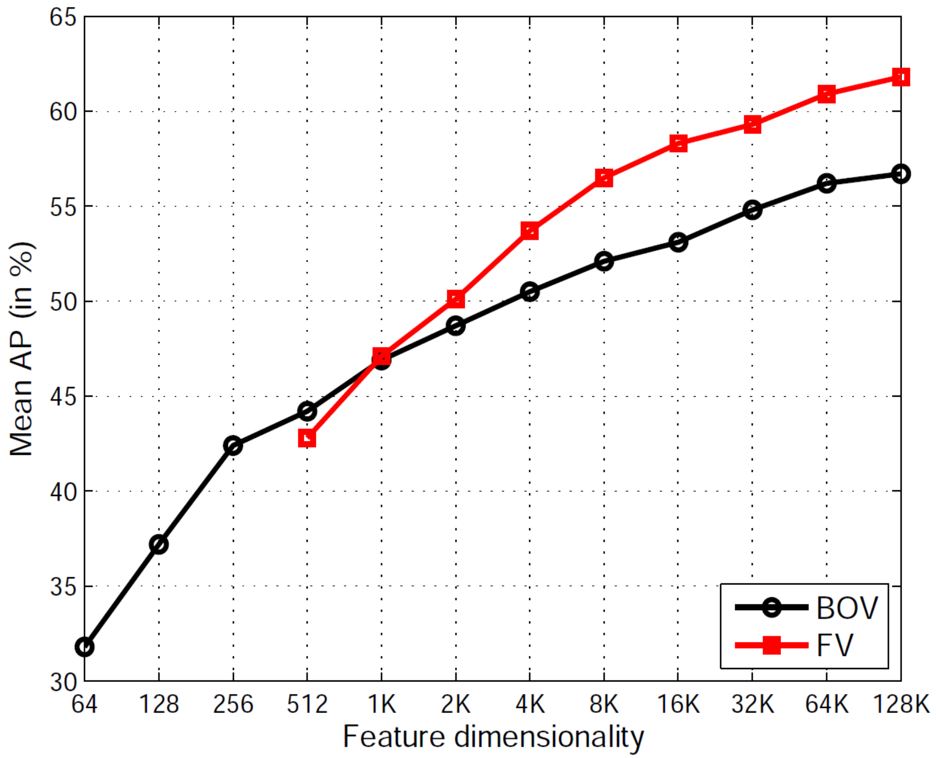
- Yearly evaluation from 2005 to 2012 for image classification

Fisher vectors: classification performance VOC'07

- Fisher vector representation yields much better performance for a given number of Gaussians / visual words than Bag-of-words.

- For a fixed dimensionality Fisher vectors
  - perform better
  - Are much cheaper to compute





Reading material

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

  Buy it if you are interested in machine learning!

- 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

Fisher vector image representation

"Fisher Kernels on Visual Vocabularies for Image Categorization"

F. Perronnin and C. Dance, in CVPR '07