Step 3: Classification

- 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

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
Functions for comparing histograms

- **L1 distance**
  \[ D(h_1, h_2) = \sum_{i=1}^{N} |h_1(i) - h_2(i)| \]

- **χ² distance**
  \[ D(h_1, h_2) = \sum_{i=1}^{N} \frac{(h_1(i) - h_2(i))^2}{h_1(i) + h_2(i)} \]

- **Quadratic distance (cross-bin)**
  \[ D(h_1, h_2) = \sum_{i,j} A_{ij} (h_1(i) - h_2(j))^2 \]
Linear classifiers

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

\[ x_i \text{ positive: } x_i \cdot w + b \geq 0 \]
\[ x_i \text{ negative: } x_i \cdot w + b < 0 \]
A **hyperplane** is defined by an equation

$$w^T x + w_0 = 0$$

- The unit vector $\frac{w}{\|w\|}$ is **normal** to the hyperplane.
- The **signed distance** of any point $x_i$ to the hyperplane is given by

$$\frac{1}{\|w\|} (w^T x_i + w_0)$$
Linear classifiers - margin

Generalization is not good in this case:

Better if a margin is introduced:
Margin maximization (for linearly separable data) is formulated as follows:

\[
\max_{(w,w_0)} M \\
\text{subject to } y_i(w^T x_i + w_0) \geq M \|w\|, \\
i = 1, \ldots, n
\]

**Explanation:** \( \frac{1}{\|w\|}(w^T x_i + w_0) \) is the signed distance between \( x_i \) and the hyperplane \( w^T x + w_0 = 0 \). The constraints require that each training point is on the correct side of the decision boundary and is at least an *unsigned* distance \( M \) from it. The goal is to find the hyperplane with parameters \( w \) and \( w_0 \) that would have the largest such \( M \).
Maximum-margin separating hyperplane

Constrained optimization problem:

$$\max_{(w, w_0)} M$$

subject to $y_i(w^T x_i + w_0) \geq M\|w\|, \quad i = 1, \ldots, n$

We can choose $M = 1/\|w\|$ and instead solve

$$\min_{(w, w_0)} \frac{1}{2}\|w\|^2$$

subject to $y_i(w^T x_i + w_0) \geq 1, \quad i = 1, \ldots, n$
Support vector machines

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

\[ x_i \text{ positive } (y_i = 1): \quad x_i \cdot w + b \geq 1 \]
\[ x_i \text{ negative } (y_i = -1): \quad x_i \cdot w + b \leq -1 \]

For support vectors, \( x_i \cdot w + b = \pm 1 \)

The margin is \( 2 / \|w\| \)
Finding the maximum margin hyperplane

1. Maximize margin \( \frac{2}{||w||} \)

2. Correctly classify all training data:
   \[
   \begin{align*}
   x_i \text{ positive } (y_i = 1): & \quad x_i \cdot w + b \geq 1 \\
   x_i \text{ negative } (y_i = -1): & \quad x_i \cdot w + b \leq -1
   \end{align*}
   \]

**Quadratic optimization problem:**

Minimize \( \frac{1}{2} w^T w \)

Subject to \( y_i (w \cdot x_i + b) \geq 1 \)

Solution based on Lagrange multipliers
Finding the maximum margin hyperplane

- Solution: \( w = \sum_i \alpha_i y_i x_i \)
Finding the maximum margin hyperplane

• Solution: \( \mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \)
  \[ b = y_i - \mathbf{w} \cdot \mathbf{x}_i \]
  for any support vector

• Classification function (decision boundary):
  \[ \mathbf{w} \cdot \mathbf{x} + b = \sum \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \]

• Notice that it relies on an inner product between the test point \( \mathbf{x} \) and the support vectors \( \mathbf{x}_i \)

• Solving the optimization problem also involves computing the inner products \( \mathbf{x}_i \cdot \mathbf{x}_j \) between all pairs of training points
Non-separable case

- What if the training data are not linearly separable? We can no longer require exact margin constraints.
- One idea: minimize

  \[
  \min_w \frac{1}{2} \|w\|^2 + C(\#\text{mistakes}).
  \]

- This is the 0-1 loss.
- The parameter \( C \) determines the penalty paid for violating margin constraints. (Tradeoff: number of mistakes and margin.)
- Problem: not QP anymore, also does not distinguish between “near misses” and bad mistakes.
Non-separable case

- Another idea: rewrite the constraints with slack variables \( \xi_i \geq 0 \):

\[
\min_{(w,w_0)} \frac{1}{2} \|w\| + C \sum_{i=1}^{n} \xi_i
\]

subject to \( y_i \left( w_0 + w^T x_i \right) - 1 + \xi_i \geq 0 \).

- Whenever margin is \( \geq 1 \) (original constraint is satisfied), \( \xi_i = 0 \).
- Whenever margin is \(< 1 \) (constraint violated), pay linear penalty.
Support vectors: points with $\alpha > 0$
- If $0 < \alpha < C$: SVs on the margin, $\xi = 0$.
- If $0 < \alpha = C$: over the margin, either misclassified ($\xi > 1$) or not ($0 < \xi \leq 1$).
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:
Nonlinear SVMs

• General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:

\[ \Phi: x \rightarrow \phi(x) \]
Nonlinear SVMs

- The kernel trick: instead of explicitly computing the lifting transformation \( \phi(x) \), define a kernel function \( K \) such that

\[
K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)
\]

(to be valid, the kernel function must satisfy Mercer’s condition)

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

\[
\sum_i \alpha_i y_i K(x_i, x) + b
\]
 Mercer’s kernels

What kind of function $K$ is a valid kernel, i.e. such that there exists a feature space $\Phi(x)$ in which $K(x, z) = \phi(x)^T \phi(z)$?

**Theorem due to Mercer (1930s)**

$K$ must be

- continuous;
- symmetric: $K(x, z) = K(z, x)$;
- positive definite: for any $x_1, \ldots, x_N$, the kernel matrix

$$K = \begin{bmatrix}
K(x_1, x_1) & K(x_1, x_2) & K(x_1, x_N) \\
& \ddots & \ddots & \ddots \\
& \ddots & \ddots & \ddots & \ddots \\
K(x_N, x_1) & K(x_N, x_2) & K(x_N, x_N) 
\end{bmatrix}$$

must be positive definite.
Radial basis function kernel

\[ K(x, z; \sigma) = \exp \left( -\frac{1}{\sigma^2} \| x - z \|^2 \right). \]

- The RBF kernel is a measure of similarity between two examples.
  - The mapping \( \phi(x) \) is infinite-dimensional!

- What is the role of parameter \( \sigma \)?
  - Consider \( \sigma \to 0 \). Then \( K(x_i, x; \sigma) \to 1 \) if \( x = z \) or \( 0 \) if \( x \neq z \).
    The SVM simply “memorizes” the training data (overfitting, lack of generalization).
  - What about \( \sigma \to \infty \)? Then \( K(x, z) \to 1 \) for all \( x, z \). The SVM underfits.
Note: some SV here not close to the boundary
Multi-class SVMs

- Various “direct” formulations exist, but they are not widely used in practice. It is more common to obtain multi-class classifiers by combining two-class SVMs in various ways.

- One vs. others:
  - Training: 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
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, \( \chi^2 \) distance, Earth Mover’s Distance, etc.
SVM classifier

SMV with multi-channel chi-square kernel

\[ K(H_i, H_j) = \exp \left( - \sum_{c \in C} \frac{1}{A_c} D_c(H_i, H_j) \right) \]

- Channel \( c \) is a combination of detector, descriptor
- \( D_c(H_i, H_j) \) is the chi-square distance between histograms
  \[ D_c(H_1, H_2) = \frac{1}{2} \sum_{i=1}^{m} \left[ \frac{(h_{1i} - h_{2i})^2}{(h_{1i} + h_{2i})} \right] \]
  \( A_c \) is the mean value of the distances between all training sample
- Extension: learning of the weights, for example with MKL

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

• Weighted sum of histogram intersections at multiple resolutions (linear in the number of features instead of cubic)

K. Grauman and T. Darrell.
Pyramid Match

Histogram intersection

\[ \mathcal{I}(H(X), H(Y)) = \sum_{j=1}^{r} \min(H(X)_j, H(Y)_j) \]

\( H(X) \)

\( H(Y) \)

\[ \mathcal{I}(H(X), H(Y)) = 4 \]
Pyramid Match

Histogram intersection

\[ \mathcal{I}(H(X), H(Y)) = \sum_{j=1}^{r} \min(H(X)_j, H(Y)_j) \]

matches at this level \( N_i = \mathcal{I}(H_i(X), H_i(Y)) - \mathcal{I}(H_{i-1}(X), H_{i-1}(Y)) \)

matches at previous level

Difference in histogram intersections across levels counts *number of new pairs* matched
Pyramid match kernel

\[
K_\Delta \left( \Psi(X), \Psi(Y) \right) = \sum_{i=0}^{L} \frac{1}{2^i} \left( \mathcal{I}(H_i(X), H_i(Y)) - \mathcal{I}(H_{i-1}(X), H_{i-1}(Y)) \right)
\]

- Weights inversely proportional to bin size
- Normalize kernel values to avoid favoring large sets
Example pyramid match

Level 0

$X$

$Y$

$H_0(X)$ $H_0(Y)$

$\mathcal{I}_0 = 2$

$N_0 = 2$

$w_0 = 1$
Example pyramid match

Level 1

$X$

$Y$

$N_1 = 4 - 2 = 2$

$w_1 = \frac{1}{2}$

$H_1(X)$

$H_1(Y)$

$I_1 = 4$
Example pyramid match

Level 2

\[ N_2 = 5 - 4 = 1 \]
\[ w_2 = \frac{1}{4} \]

\[ H_2(X) \]
\[ H_2(Y) \]
\[ I_2 = 5 \]
Example pyramid match

\[ K_\Delta = \sum_{i=0}^{L} w_i N_i \]

\[ = 1(2) + \frac{1}{2}(2) + \frac{1}{4}(1) = 3.25 \]

\[ K = \max_{\pi: X \rightarrow Y} \sum_{x_i \in X} S(x_i, \pi(x_i)) \]

\[ = 1(2) + \frac{1}{2}(3) = 3.5 \]
Summary: Pyramid match kernel

\[ K_\Delta (\Psi(X), \Psi(Y)) = \sum_{i=0}^{L} \frac{1}{2^i} \left( \mathcal{I}(H_i(X), H_i(Y)) - \mathcal{I}(H_{i-1}(X), H_{i-1}(Y)) \right) \]

difficulty of a match at level \( i \)  
number of new matches at level \( i \)

optimal partial matching between sets of features
Review: Discriminative methods

- Nearest-neighbor and k-nearest-neighbor classifiers
  - L1 distance, $\chi^2$ distance, quadratic distance,

- Support vector machines
  - Linear classifiers
  - Margin maximization
  - The kernel trick
  - Kernel functions: histogram intersection, generalized Gaussian, pyramid match

- Of course, there are many other classifiers out there
  - Neural networks, boosting, decision trees, …
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
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
Generative methods

• Model the probability distribution that produced a given bag of features

• We will cover two models, both inspired by text document analysis:
  • Naïve Bayes
  • Probabilistic Latent Semantic Analysis
The Naïve Bayes model

- Assume that each feature is conditionally independent \textit{given the class}

\[
p(w_1, \ldots, w_N \mid c) = \prod_{i=1}^{N} p(w_i \mid c)
\]
The Naïve Bayes model

• Assume that each feature is conditionally independent given the class

\[
c^* = \arg \max_c \ p(c) \prod_{i=1}^{N} p(w_i \mid c)
\]

MAP decision
Prior prob. of the object classes
Likelihood of ith visual word given the class

Estimated by empirical frequencies of visual words in images from a given class

Csurka et al. 2004
The Naïve Bayes model

- Assume that each feature is conditionally independent given the class

\[ c^* = \arg \max_c p(c) \prod_{i=1}^{N} p(w_i | c) \]

- “Graphical model”:
Probabilistic Latent Semantic Analysis

zebra

grass

tree

“visual topics”

T. Hofmann, Probabilistic Latent Semantic Analysis, UAI 1999
Probabilistic Latent Semantic Analysis

New image = $\alpha_1$ + $\alpha_2$ + $\alpha_3$

T. Hofmann, Probabilistic Latent Semantic Analysis, UAI 1999
• Unsupervised technique
• Two-level generative model: a document is a mixture of topics, and each topic has its own characteristic word distribution

T. Hofmann, Probabilistic Latent Semantic Analysis, UAI 1999
Probabilistic Latent Semantic Analysis

- Unsupervised technique
- Two-level generative model: a document is a mixture of topics, and each topic has its own characteristic word distribution

\[ p(w_i \mid d_j) = \sum_{k=1}^{K} p(w_i \mid z_k) p(z_k \mid d_j) \]

T. Hofmann, Probabilistic Latent Semantic Analysis, UAI 1999
pLSA for images

Document = image, topic = class, word = quantized feature

J. Sivic, B. Russell, A. Efros, A. Zisserman, B. Freeman,
Discovering Objects and their Location in Images, ICCV 2005
The pLSA model

\[ p(w_i \mid d_j) = \sum_{k=1}^{K} p(w_i \mid z_k) p(z_k \mid d_j) \]

- Probability of word \( i \) in document \( j \) (known)
- Probability of word \( i \) given topic \( k \) (unknown)
- Probability of topic \( k \) given document \( j \) (unknown)
The pLSA model

\[ p(w_i \mid d_j) = \sum_{k=1}^{K} p(w_i \mid z_k) p(z_k \mid d_j) \]

Observed codeword distributions \((M \times N)\)

Codeword distributions per topic (class) \((M \times K)\)

Class distributions per image \((K \times N)\)
Learning pLSA parameters

Maximize likelihood of data using EM:

\[ L = \prod_{i=1}^{M} \prod_{j=1}^{N} P(w_i|d_j)^{n(w_i,d_j)} \]

- \( M \) … number of codewords
- \( N \) … number of images

Observed counts of word \( i \) in document \( j \)

\[ \sum_{k=1}^{K} P(z_k|d_j) P(w_i|z_k) \]
Recognition

• Finding the most likely topic (class) for an image:

\[ z^* = \arg \max_z p(z \mid d) \]
Recognition

- Finding the most likely topic (class) for an image:

\[ z^* = \arg \max_z p(z \mid d) \]

- Finding the most likely topic (class) for a visual word in a given image:

\[ z^* = \arg \max_z p(z \mid w, d) = \arg \max_z \frac{p(w \mid z)p(z \mid d)}{\sum_{z'} p(w \mid z')p(z' \mid d)} \]
Topic discovery in images

J. Sivic, B. Russell, A. Efros, A. Zisserman, B. Freeman,
Discovering Objects and their Location in Images, ICCV 2005
Summary: Generative models

• Naïve Bayes
  • Unigram models in document analysis
  • Assumes conditional independence of words given class
  • Parameter estimation: frequency counting

• Probabilistic Latent Semantic Analysis
  • Unsupervised technique
  • Each document is a mixture of topics (image is a mixture of classes)
  • Can be thought of as matrix decomposition
  • Parameter estimation: EM