Lecture 8
Recognition and Classification
(continued)
Review: Image Classification Steps

**Training**
- Training Images
  - [Image of various fruits and animals]
- Training Labels
- Image Features
- Training
- Learned model

**Testing**
- Test Image
- Image Features
- Learned model
- Prediction

**Notes:**
- D. Hoiem and L. Lazebnik
Review: Object Detection Typical Components

- **Hypothesis** generation
  - Sliding window, Segmentation, feature point detection, random, search

- **Encoding** of (local) image data
  - Colors, Edges, Corners, Histogram of Oriented Gradients, Wavelets, Convolution Filters

- **Relationship** of different parts to each other
  - Blur or histogram, Tree/Star, Pairwise/Covariance

- **Learning** from (labeled) examples
  - Selecting representative examples (templates), Clustering, Building a cascade
  - Classifiers: Bayes, Logistic regression, SVM, AdaBoost, Neural Network...
  - Generative vs. Discriminative

- **Verification** - removing redundant, overlapping, incompatible examples
  - Non-Max Suppression, context priors, geometry
Example 1: Chamfer matching (Pedestrian Detection)

Input Image  Edge Detection  Template  Find Best Match

\[ D_{\text{chamfer}}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t) \]

Gavrila & Philomin ICCV 1999

Slides from K. Grauman and B. Leibe
Example 1: Chamfer matching (Pedestrian Detection)

Hierarchy of templates

Gavrila & Philomin ICCV 1999
Clustering Strategies

- **K-means**
  - Iteratively re-assign points to the nearest cluster center

- **Mean-shift clustering**
  - Estimate modes of pdf

- **Agglomerative clustering**
  - Start with each point as its own cluster and iteratively merge the closest clusters

- **Spectral clustering**
  - Split the nodes in a graph based on assigned links with similarity weights

As we go down this chart, the clustering strategies have more tendency to transitively group points even if they are not nearby in feature space.
Example 2: Viola/Jones (Face Detection)

Features: “Haar-like Rectangle filters”

- Differences between sums of pixels in adjacent rectangles

2-rectangle features

3-rectangle features

4-rectangle features

$60,000 \times 100 = 6,000,000$

Unique Features
Example 2: Viola/Jones - Integral Images

- \( ii = \text{cumsum}(\text{cumsum}(\text{im}, 1), 2) \)

\( ii(x,y) = \) Sum of the values in the grey region

How to compute \( A+D-B-C \)?

How to compute \( B-A \)?

How to compute \( A+D-B-C \)?
Example 2: Feature selection with Adaboost

1. Create a large pool of features
2. Select features that are discriminative and work well together:
   - “Weak learner” = feature + threshold + parity
   - Choose weak learner that minimizes error on the weighted training set
   - Reweight

(More on AdaBoost next time...)
Example 2: Viola/Jones Cascaded Classifier

- first classifier: 100% detection, 50% false positives.
- second classifier: 100% detection, 40% false positives (20% cumulative) using data from previous stage.
- third classifier: 100% detection, 10% false positive rate (2% cumulative)

- Put cheaper classifiers up front
Example 2: Viola/Jones results

Run-time: 15fps  (384x288 pixel image on a 700 Mhz Pentium III)
Typical Components

- **Hypothesis** generation
  - Whole image, Sliding window, Segmentation, Feature point detection, Search...

- **Encoding** of (local) image data
  - Colors, Edges, Corners, Histogram of Oriented Gradients, Wavelets, Convolution Filters...

- **Relationship** of different parts to each other
  - Blur or histogram, Tree/Star, Pairwise/Covariance...

- **Learning** from labeled examples
  - Selecting representative examples (templates), Clustering, Building a cascade
  - Classifiers: Bayes, Logistic regression, SVM, AdaBoost, ...
  - Generative vs. Discriminative

- **Verification** - removing redundant, overlapping, incompatible examples
  - Non-Max Suppression, context priors, geometry

Geometry is Hard. Let's ignore it...
(No Geometry) Example: Bag of Words

Object → Bag of ‘words’

Slide Credit: Svetlana Lazebnik
Objects as texture

- All of these are treated as being the same

- No distinction between foreground and background: scene recognition?
Origin 1: Texture recognition

- Texture is characterized by the repetition of basic elements or *textons*
- For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary

Salton & McGill (1983)
• Interest Point Features

\[ \text{( ) ( ) ( ) ( ) ...} \]
Clustering

Vector quantization
Recall: SIFT Matching with RANSAC

SIFT + RANSAC transformation matching to exemplars was not likely to work

How is SIFT-based classification through Bag of Words different?
The (obvious) problem with ignoring Geometry

All of these images have the same color histogram
Adding Geometry back: Spatial pyramid

Compute histogram in each spatial bin
Spatial pyramid representation

- Extension of a bag of features
- Locally orderless representation at several levels of resolution

Lazebnik, Schmid & Ponce (CVPR 2006)
Object Classification technique: Delal & Triggs

- **Hypothesis** generation
  - Sliding window, Segmentation, feature point detection, random, search

- **Encoding** of (local) image data
  - Colors, Edges, Corners, Histogram of Oriented Gradients, Wavelets, Convolution Filters

- **Relationship** of different parts to each other
  - Blur or histogram, Tree/Star, Pairwise/Covariance

- **Learning** from labeled examples
  - Selecting representative examples (templates), Clustering, Building a cascade
  - Classifiers: K-NN, Logistic regression, SVM, AdaBoost, Bayes ...
  - Generative vs. Discriminative

- **Verification** - removing redundant, overlapping, incompatible examples
  - Non-Max Suppression, context priors, geometry
In practice, effect is very small (about 1%) while some computational time is required*

Computing gradients

<table>
<thead>
<tr>
<th>Mask Type</th>
<th>1D centered</th>
<th>1D uncentered</th>
<th>1D cubic,corrected</th>
<th>2x2 diagonal</th>
<th>3x3 Sobel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
<td>[-1, 0, 1]</td>
<td>[-1, 1]</td>
<td>[1, -8, 0, 8, -1]</td>
<td>[0 1]</td>
<td>[-1 0 1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[-2 0 2]</td>
<td>[-1 0 1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[-1 0]</td>
<td>[-1 -2 -1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0 0 0]</td>
<td>[1 2 1]</td>
</tr>
<tr>
<td>Miss rate at 10^{-4} FPPW</td>
<td>11%</td>
<td>12.5%</td>
<td>12%</td>
<td>12.5%</td>
<td>14%</td>
</tr>
</tbody>
</table>
Histogram of Oriented Gradients

[Dalal and Triggs, CVPR 2005]

Slide Credit: Noah Snavely
Histogram of Oriented Gradients
HOG feature vector for one block

\[ f = (h_1^1, ..., h_1^9, h_2^1, ..., h_2^9, h_3^1, ..., h_3^9, h_4^1, ..., h_4^9) \]

- **Angle**
  - 0: 15, 25, 25
  - 10: 15, 25, 30
  - 45: 95, 101, 110
  - 47: 97, 101, 120

- **Magnitude**
  - 5: 20, 20, 10
  - 10: 10, 10, 5
  - 20: 30, 30, 40
  - 50: 70, 70, 80

Feature vector extends while window moves.
- Accumulate weight votes over spatial cells

- How many bins should be in histogram?
- Should we use oriented or non-oriented gradients?
- How to select weights?
- Should we use overlapped blocks or not? If yes, then how big should be the overlap?
- What block size should we use?
Accumulate weight votes over spatial cells

- How many bins should be in histogram?
- Should we use oriented or non-oriented gradients?
- How to select weights?
- Should we use overlapped blocks or not? If yes, then how big should be the overlap?
- What block size should we use?
Accumulate weight votes over spatial cells

- How many bins should be in histogram?
- Should we use oriented or non-oriented gradients?
- How to select weights?
- Should we use overlapped blocks or not? If yes, then how big should be the overlap?
- What block size should we use?

![Graph showing DET curves with different overlap and stride configurations.](image)
Contrast normalization

\[ L_1 - \text{norm} = \frac{v}{\|v\|_1 + \varepsilon} \quad L_1 - \text{sqrt} = \sqrt{\frac{v}{\|v\|_1 + \varepsilon}} \quad L_2 - \text{norm} = \frac{v}{\sqrt{\|v\|_2^2 + \varepsilon}} \]

\[ L_2 - \text{Hys} - \text{L2-norm followed by clipping (limiting the maximum values of } v \text{ to 0.2) and renormalising} \]
Making the feature vector

Input image

- Detection window

1. Normalise gamma & colour
2. Compute gradients
3. Accumulate weighted votes for gradient orientation over spatial cells
4. Normalise contrast within overlapping blocks of cells
5. Collect HOGs for all blocks over detection window

Variants of HOG descriptors. (a) A rectangular HOG (R-HOG) descriptor with $3 \times 3$ blocks of cells. (b) Circular HOG (C-HOG) descriptor with the central cell divided into angular sectors as in shape contexts. (c) A C-HOG descriptor with a single central cell.
In each triplet: (1) the input image, (2) the corresponding R-HOG feature vector (only the dominant orientation of each cell is shown), (3) the dominant orientations selected by the SVM (obtained by multiplying the feature vector by the corresponding weights from the linear SVM).
Feature Vector → Classification Result

<table>
<thead>
<tr>
<th>Supervised Learning</th>
<th>Unsupervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete</td>
<td></td>
</tr>
<tr>
<td>classification or categorization</td>
<td>clustering</td>
</tr>
<tr>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>regression</td>
<td>dimensionality reduction</td>
</tr>
</tbody>
</table>
The machine learning framework

\[ \hat{y} = f(x) \]

- **Training**: given a *training set* of labeled examples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \), estimate the prediction function \( f \) by minimizing the prediction error on the training set.
- **Testing**: apply \( f \) to a never before seen *test example* \( x \) and output the predicted value \( y = f(x) \).
Classification

• Assign input vector to one of two or more classes

• Any decision rule divides input space into decision regions separated by decision boundaries
Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- Etc.

Which is the best one?
Which Algorithm to use?

**Classification**
- SVC
- Ensemble Classifiers
- Naive Bayes
- Text Data
- Linear SVC
- Kernel approximation
- NOT WORKING

**Clustering**
- Spectral Clustering
- GMM
- KMeans
- MiniBatch KMeans
- MeanShift
- VBGMM
- <10K samples
- <10K categories known
- <10K samples
- Out of luck
- Predicting a category
- Predicting a quantity
- Predicting structure
- Randomized PCA
- Isomap
- Spectral Embedding
- LLE
- <10K samples
- <10K samples
- Not working
- Not working
- Not working
- Not working
- Not working

**Regression**
- SGD Regressor
- Lasso
- ElasticNet
- SVR(kernel="rbf")
- Ensemble Regressors
- Ridge Regression
- SVR(kernel="linear")
- <100K samples
- <100K samples
- few features should be important
- NOT WORKING
- NOT WORKING
- NOT WORKING
- NOT WORKING
- NOT WORKING
- NOT WORKING

**Dimensionality Reduction**
- Kernel approximation
- <10K samples
- <10K samples
- Not working
- Not working
- Not working
- Not working
- Not working
- Not working
- NOT WORKING
- NOT WORKING
- NOT WORKING
- NOT WORKING

START
(1-) Nearest Neighbor

\[ f(x) = \text{label of the training example nearest to } x \]

- All we need is a distance function for our inputs
- No training required!
K-nearest neighbor
1-nearest neighbor
3-nearest neighbor
5-nearest neighbor
Questions:
- What distance function to use L1, L2?
- What is the accuracy of the 1-NN classifier on the training data?
- What is the accuracy of the 5-NN classifier on the training data?
- Which one do you expect to do better on the test data?
Classifiers: Linear

- Find a *linear function* to separate the classes:

\[ \hat{y} = f(x) = \text{sgn}(w \cdot x + b) \]
Linear classifiers

- Find linear function to separate positive and negative examples

\[
x_i \text{ positive: } x_i \cdot w + b \geq 0
\]

\[
x_i \text{ negative: } x_i \cdot w + b < 0
\]

Which line is best?
Using Least Squares for Classification

- Find a linear function to separate the classes:

\[ \hat{y} = f(x) = \text{sgn}(w \cdot x + b) \]
Using Least Squares for Classification

- Find a linear function to separate the classes:

\[ \hat{y} = f(x) = \text{sgn}(w \cdot x + b) \]
Using least squares for classification

If the right answer is 1 and the model says 1.5, it loses, so it changes the boundary to avoid being “too correct”
The Problem: Loss Function

Recall: Regression Loss Functions

Some Classification Loss Functions

Squared Loss
Hinge Loss
Log Loss
0/1 Loss
Sigmoid

We model the probability of a label $Y$ to be equal $y \in \{-1, 1\}$, given a data point $x \in \mathbb{R}^n$, as:

$$P(Y = y \mid x) = \frac{1}{1 + \exp(-y(w^T x + b))}.$$ 

This amounts to modeling the log-odds ratio as a linear function of $X$:

$$\log \frac{P(Y = 1 \mid x)}{P(Y = -1 \mid x)} = w^T x + b.$$

- The decision boundary $P(Y = 1 \mid x) = P(Y = -1 \mid x)$ is the hyperplane with equation $w^T x + b = 0$.
- The region $P(Y = 1 \mid x) \geq P(Y = -1 \mid x)$ (i.e., $w^T x + b \geq 0$) corresponds to points with predicted label $\hat{y} = +1$. 
The likelihood function is

\[ l(w, b) = \prod_{i=1}^{m} \frac{1}{1 + e^{-y_i(w^T x_i + b)}}. \]

Now maximize the log-likelihood:

\[ \max_{w, b} L(w, b) := -\sum_{i=1}^{m} \log(1 + e^{-y_i(w^T x_i + b)}) \]

In practice, we may consider adding a regularization term

\[ \max_{w, b} L(w, b) + \lambda r(w), \]

with \( r(w) = \|w\|_2^2 \) or \( r(x) = \|w\|_1 \).
Logistic Result

Logistic Regression

least squares regression
Using Logistic Regression

- Quick, simple classifier (try it first)
- Outputs a probabilistic label confidence
- Use L2 or L1 regularization
  - L1 does feature selection and is robust to irrelevant features but slower to train
• Find a *linear function* to separate the classes:

$$f(x) = \text{sgn}(w \cdot x + b)$$
Classifiers: Linear SVM

• Find a linear function to separate the classes:

\[ f(x) = \text{sgn}(w \cdot x + b) \]
• Find a **linear function** to separate the classes:

\[
f(x) = \text{sgn}(w \cdot x + b)
\]
Support vector machines: Margin

- Want line that maximizes the margin.

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

For support vectors, $x_i \cdot (y_i = 1): x_i \cdot w + b \geq 1$

For support vectors, $x_i \cdot (y_i = -1): x_i \cdot w + b \leq -1$

Hinge Loss

$L(y, f(x)) = \max(0, 1 - y \cdot f(x))$

C. Burges,
A Tutorial on Support Vector Machines for Pattern Recognition

Support vectors
Margin

C. Burges,
A Tutorial on Support Vector Machines for Pattern Recognition
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:

\[ \Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x}) \]
Nonlinear SVMs

• *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(x)$, define a kernel function $K$ such that

$$K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$$

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

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

$$\sum_i \alpha_i y_i \varphi(x_i) \cdot \varphi(x) + b = \sum_i \alpha_i y_i K(x_i, x) + b$$

Nonlinear kernel: Example

• Consider the mapping 
\[ \phi(x) = (x, x^2) \]

\[ \phi(x) \cdot \phi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2 \]

\[ K(x, y) = xy + x^2 y^2 \]
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 (inverse) L1 distance, Euclidean distance, \(\chi^2\) distance, etc.

What about multi-class SVMs?

• Unfortunately, there is no “definitive” multi-class SVM formulation

• In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs

• 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
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
Spectrum of supervision

Unsupervised

“Weakly” supervised

Fully supervised

Definition depends on task
Generalization

- How well does a learned model generalize from the data it was trained on to a new test set?
Train vs. Test Accuracy

- **Test Set**
- **Training Set**

Error vs. Model Complexity graph with the optimum model complexity between the training and test set error rates.
Generalization

- **Components of generalization error**
  - **Bias**: how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - **Variance**: how much models estimated from different training sets differ from each other

- **Underfitting**: model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error

- **Overfitting**: model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error
Bias-Variance Trade-off

- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).

- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).
Bias-Variance Trade-off

\[ E(\text{MSE}) = \text{noise}^2 + \text{bias}^2 + \text{variance} \]

- Unavoidable error
- Error due to incorrect assumptions
- Error due to variance of training samples

See the following for explanations of bias-variance (also Bishop’s “Neural Networks” book):
- [http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf](http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf)
Try out what hyperparameters work best on test set.
Trying out what hyperparameters work best on test set: Very bad idea. The test set is a proxy for the generalization performance! Use only VERY SPARINGLY, at the end.
Validation data
use to tune hyperparameters
Cross-validation cycle through the choice of which fold is the validation fold, average results.
Remember...

- No classifier is inherently better than any other: you need to make assumptions to generalize

- Three kinds of error
  - Inherent: unavoidable
  - Bias: due to over-simplifications
  - Variance: due to inability to perfectly estimate parameters from limited data
What to remember about classifiers

- Machine learning algorithms are tools, not dogmas
- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)
How to reduce variance?

- Choose a simpler classifier
- Regularize the parameters
- Get more training data
### Generative vs. Discriminative Classifiers

**Generative Models**
- Represent both the data and the labels
- Often, makes use of conditional independence and priors
- Examples
  - Naïve Bayes classifier
  - Bayesian network

- Models of data may apply to future prediction problems

**Discriminative Models**
- Learn to directly predict the labels from the data
- Often, assume a simple boundary (e.g., linear)
- Examples
  - Logistic regression
  - SVM
  - Boosted decision trees

- Often easier to predict a label from the data than to model the data
Some Machine Learning References

- **General**
  - Christopher Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995

- **Adaboost**

- **SVMs**