Review: Image Classification Steps

Training

- Training Images

- Training Labels

- Training

- Learned model

Testing

- Test Image

- Image Features

- Learned model

- Prediction

D. Hoiem and L. Lazebnik

Slide Credit: D. Hoiem and L. Lazebnik
Review: 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, ...
  - Generative vs. Discriminative

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

Hierarchy of templates

Gavrila & Philomin ICCV 1999
Example 2: Viola/Jones (Face Detection)

Robust Realtime Face Detection, IJCV 2004, Viola and Jones

Features: “Haar-like Rectangle filters”

- Differences between sums of pixels in adjacent rectangles

- 2-rectangle features
- 3-rectangle features
- 4-rectangle features

Unique Features: $60,000 \times 100 = 6,000,000$

Slide from: Derek Hoiem
(No Geometry) Example: Color Histograms

(No Geometry) Example: Bag of Words

Object

Bag of ‘words’
Clustering (usually k-means)

Vector quantization
Machine Learning Problems

- **Supervised Learning**
  - Discrete: classification or categorization
  - Continuous: regression

- **Unsupervised Learning**
  - clustering
  - dimensionality reduction

Slide Credit: James Hayes
Clustering Strategies

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

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

• Mean-shift clustering
  - Estimate modes of pdf

• 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
Today

- **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
Dalal and Trigs Pedestrian Detector

In practice, effect is very small (about 1%) while some computational time is required.*

### Computing gradients

#### Input image
- Detection window

#### Processing steps
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

#### 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></td>
<td></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>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Slide Credit: Qing J. Wang & Ru B. Zhang
Histogram of Oriented Gradients

HoGify

10x10 cells

20x20 cells

[Dalal and Triggs, CVPR 2005]
Histogram of Oriented Gradients
HOG feature vector for one block

\[ f = (h_1^1, ..., h_9^1, h_1^2, ..., h_9^2, h_1^3, ..., h_9^3, h_1^4, ..., h_9^4) \]

<table>
<thead>
<tr>
<th>Angle</th>
<th>Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>25</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td>45</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>110</td>
</tr>
<tr>
<td>47</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>120</td>
</tr>
</tbody>
</table>

Binary voting

<table>
<thead>
<tr>
<th>Magnitude voting</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
</tr>
<tr>
<td>400</td>
</tr>
<tr>
<td>300</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>

Feature vector extends while window moves

Slide Credit: Qing J. Wang & Ru B. Zhang
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?

Input image

Detection window

Normalise gamma & colour

Compute gradients

Accumulate weighted votes for gradient orientation over spatial cells

Normalise contrast within overlapping blocks of cells

Collect HOGs for all blocks over detection window

![Graph showing DET - effect of overlap (cell size=8, num cell = 2x2, wt=0)]
Contrast normalization

\[ L1\-\text{norm} = \frac{v}{\|v\| + \varepsilon} \quad L1\-\text{sqrt} = \frac{v}{\sqrt{\|v\| + \varepsilon}} \quad L2\-\text{norm} = \frac{v}{\sqrt{\|v\|^2 + \varepsilon}} \]

\(L2\-\text{Hys}\) - L2-norm followed by clipping (limiting the maximum values of \(v\) to 0.2) and renormalising

Input image
- Detection window

Normalise gamma & colour

Compute gradients

Accumulate weighted votes for gradient orientation over spatial cells

Normalise contrast within overlapping blocks of cells

Collect HOGs for all blocks over detection window
Making feature vector

Input image

Detection window

Normalise gamma & colour

Compute gradients

Accumulate weighted votes for gradient orientation over spatial cells

Normalise contrast within overlapping blocks of cells

Collect HOGs for all blocks over detection window

(a) R-HOG/SIFT
(b) C-HOG
(c) Single centre C-HOG

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 $\rightarrow$ Classification Result

- **Supervised Learning**
  - Discrete: classification or categorization
  - Continuous: regression

- **Unsupervised Learning**
  - Continuous: dimensionality reduction
  - Discrete: clustering

Slide Credit: James Hayes
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
- KNeighbors Classifier
- SGD Classifier
- kernel approximation

<100K samples
- do you have labeled data?
  - yes: get more data
  - no: predicting a category
    - >50 samples
      - yes: SGD Regressor
      - no: regression
        - few features should be important
          - yes: Lasso, ElasticNet
          - no: SVR(kernel="linear")
          - not working: RidgeRegression
          - not working: SVR(kernel="rbf")
    - no: predicting a quantity
      - yes: <100K samples
        - few features should be important
        - yes: SGD Regressor
        - no: regression
          - few features should be important
          - yes: Lasso, ElasticNet
          - no: SVR(kernel="linear")
          - not working: RidgeRegression
          - not working: SVR(kernel="rbf")
      - no: dimensionality reduction
        - Randomized PCA
        - Isomap
        - Spectral Embedding
        - kernel approximation
        - not working: LLE

<10K samples
- number of categories known
  - yes: MiniBatch KMeans
  - no: MeanShift

<10K samples
- looking
  - yes: predicting structure
  - no: tough luck

clustering
- Spectral Clustering
- GMM
- KMeans
- MeanShift
- VBGMM

back
(1-) Nearest Neighbor

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

- 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

![Diagram of 5-nearest neighbor with data points and circles.]
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 expect to do better on the test data?
• 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
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 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:

\[ f(x) = \text{sgn}(w \cdot x + b) \]
• 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.

\[ w \cdot x + b = 1 \]
\[ w \cdot x + b = 0 \]
\[ w \cdot x + b = -1 \]

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

\[
\begin{align*}
\text{For positive } (y_i = 1): & \quad x_i \cdot w + b \geq 1 \\
\text{For negative } (y_i = -1): & \quad x_i \cdot w + b \leq -1
\end{align*}
\]

Hinge Loss

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

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: 

\[ \Phi: x \rightarrow \varphi(x) \]
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:

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

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