Overview

• Recall last class: Boosting is a way of generating a strong classifier as a weighted ensemble of weak ones
• Today: Support Vector Machine (SVM) training generates a strong classifier directly
• Case Study: Dalal and Triggs pedestrian detector

Support Vector Machines

SVM slides from Kristen Grauman, UT-Austin

Other good resources:

- Presentation slides from Christoph Lampert: https://sites.google.com/site/christophlam/teaching/kernel-methods-for-object-recognition
- Simple tutorial document by Chris Williams: http://www.inf.ed.ac.uk/teaching/courses/iam/docs/svm.pdf
- Video lecture by Pat Winston: https://www.youtube.com/watch?v=_PwhWxkKBo

Linear classifiers

Find linear function to separate positive and negative examples

\[
\begin{align*}
\text{Let} \quad w &= \begin{bmatrix} a \\ c \end{bmatrix} \quad x &= \begin{bmatrix} x \\ y \end{bmatrix} \\
ax + cy + b &= 0 \\
w \cdot x + b &= 0
\end{align*}
\]

Linear classifiers

Find linear function to separate positive and negative examples

\[
\begin{align*}
\text{Let} \quad w &= \begin{bmatrix} a \\ c \end{bmatrix} \quad x &= \begin{bmatrix} x \\ y \end{bmatrix} \\
ax + cy + b &= 0 \\
w \cdot x + b &= 0
\end{align*}
\]

Which line is best?
Support Vector Machines (SVMs)

- Discriminative classifier based on optimal separating line (for 2d case)
- Maximize the margin between the positive and negative training examples

\[
\begin{align*}
\text{Lines in } \mathbb{R}^2 & : \quad w = [a, c] \quad x = [x, y] \\
ax + cy + b = 0 \quad & \quad w \cdot x + b = 0 \\
D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} \quad & \quad \text{distance from point to line}
\end{align*}
\]

\[
\begin{align*}
\text{Support vector machines} & : \quad \text{Want line that maximizes the margin.} \\
\text{x, positive } (y = 1) : & \quad x \cdot w + b > 1 \\
\text{x, negative } (y = -1) : & \quad x \cdot w + b < -1 \\
\text{For support vectors, } & \quad x_i \cdot w + b = \pm 1 \\
\text{Distance between point and line: } & \quad \frac{|x \cdot w + b|}{\|w\|} \quad \text{M} = \frac{1}{2}
\end{align*}
\]
Support vector machines

• Want line that maximizes the margin.

Support vectors

Margin

Finding the maximum margin line

1. Maximize margin $2/\|w\|$
2. Correctly classify all training data points:
   - $x_i$ positive ($y_i = 1$): $x_i \cdot w + b \ge 1$
   - $x_i$ negative ($y_i = -1$): $x_i \cdot w + b \le -1$

Quadrahtic optimization problem:

$$\min \frac{1}{2} w^T w$$
$$\text{Subject to } y_i (w \cdot x_i + b) \ge 1$$

One constraint for each training point.
Note sign trick.

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

Distance between point and line:

$$\frac{|x_i \cdot w + b|}{\|w\|}$$

Therefore, the margin is $2 / \|w\|$

Finding the maximum margin line

• Solution: $w = \sum \alpha_i y_i x_i$

Learned weight
Support vector

Questions

• What if the features are not 2d?
• What if the data is not linearly separable?
• What to do for more than two classes?

Questions

• What if the features are not 2d?
  – Generalizes to d-dimensions – replace line with "hyperplane"
• What if the data is not linearly separable?
• What to do for more than two classes?
Planes in $\mathbb{R}^3$

Let $w = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ and $x = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

$$ax + by + cz + d = 0$$

$$w \cdot x + d = 0$$

$$D = \frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}} = \frac{w^T x + d}{\|w\|}$$ distance from point to plane

Hyperplanes in $\mathbb{R}^n$

Hyperplane $H$ is set of all vectors $x \in \mathbb{R}^n$ which satisfy:

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n + b = 0$$

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

$$D(H, x) = \frac{w^T x + b}{\|w\|}$$ distance from point to hyperplane

Questions

- What if the features are not 2d?
- What if the data is not linearly separable?

Nonlinear SVMs

Solution 1: use polar coordinates

* Data is linearly separable in polar coordinates
* Acts non-linearly in original space

$$\Phi: \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} r \\ \theta \end{bmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

Nonlinear SVMs

Solution 2: map data to higher dimension

$$\Phi: \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

* Data is linearly separable in 3D
* This means that the problem can still be solved by a linear classifier

Nonlinear SVMs

SVM classifiers in a transformed feature space

$$f(x) = 0$$

$$\Phi: x \rightarrow \Phi(x) \quad \mathbb{R}^D \rightarrow \mathbb{R}^D$$

Learn classifier linear in $w$ for $\mathbb{R}^D$:

$$f(x) = w^T \Phi(x) + b$$

$\Phi(x)$ is a feature map
The Kernel Trick

- Recall we transformed linear regression into nonlinear regression using a feature vector $\Phi(x)$ and, ultimately, the “kernel trick.”

- We also use the kernel trick here to transform a linear classifier into nonlinear one.

Example Kernel

$$
\Phi : \left( \begin{array}{c} x_1 \\ x_2 \\ \end{array} \right) \rightarrow \left( \begin{array}{c} x_1^2 \\ \sqrt{x_1 x_2} \\ \end{array} \right) \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3 \\
\Phi(x)^T \Phi(x) = (x_1^2 + \sqrt{x_1 x_2}) (x_1^2 + \sqrt{x_1 x_2}) \\
= x_1^4 + 2x_1^2 \sqrt{x_1 x_2} + 2x_1 \sqrt{x_1 x_2} + x_2^2 \\
= (x^T a)^2
$$

Kernel Trick

- Classifier can be kernel and applied without explicitly computing $\Phi(x)$
- All that is required is the kernel $k(x, y) = (x^T y)^2$
- Complexity of learning depends on $N$ (typically it is $O(N^2)$) not on $D$

Example Kernels

- **Linear kernels** $k(x, x') = x^T x'$
- **Polynomial kernels** $k(x, x') = (1 + x^T x')^d$ for any $d > 0$
  - Contains all polynomials terms up to degree $d$
- **Gaussian kernels** $k(x, x') = \exp\left(-\frac{\|x - x\|^2}{2\sigma^2}\right)$ for $\sigma > 0$
  - Infinite dimensional feature space

Nonlinear SVMs

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

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

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

$$
\sum_{i=1}^{N} r_i y_i K(x_i, x) + b
$$

Questions

- What if the features are not 2d?
- What if the data is not linearly separable?
- What to do for more than two classes?

Multi-class SVMs

- Achieve multi-class classifier by combining a number of binary classifiers

  - **One vs. all**
    - Training: learn an SVM for each class vs. the rest
    - 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

\[\text{Slide from Andrew Zisserman}\]
Software for SVMs
There exist many implementations of maximum margin classification with and without support for arbitrary kernel functions:

- General purpose SVMs packages: LibSVM\(^1\), SVMlight\(^2\)
  - source code, applications, wrappers for Matlab, Python, R...
  - standard kernels are included, user kernels can be defined
  - sparse data format (sometimes annoying for vision data)

- Very fast implementations for linear kernels: SVMlin\(^3\), SGD\(^4\)

- Larger toolboxes with wrappers for SVMlight or LibSVM:
  MLPy, Orange, PyML, Shogun, Spider, Torch, YALE, Weka...

\(^1\)http://www.csie.ntu.edu.tw/~cjlin/libsvm/
\(^2\)http://svmlight.joachims.org/
\(^3\)http://people.cs.uchicago.edu/~riikne/svmlin.html
\(^4\)http://leon.bottou.org/projects/sgd

SVMs for recognition
1. Define a vector representation for each example.
2. Select a kernel function.
3. Compute pairwise kernel values between labeled examples
4. Given this “kernel matrix” to SVM optimization software to identify support vectors & weights.
5. To classify a new example: compute kernel values between new input and support vectors, apply weights, check sign of output.

Case Study: Pedestrian Detector
Navneet Dalal and Bill Triggs, “Histograms of Oriented Gradients for Human Detection,” CVPR 2005

- Detect upright pedestrians
- Histogram of oriented gradient feature vector
- Linear SVM classifier; sliding window detector

\[
x_i \in \mathbb{R}^d, \text{ with } d = 1024
\]

HoG Feature Extraction:

- Input image
- Detection window
- Normalise gamma
- Compute gradients
- Weighted vote in spatial & orientation cells
- Contrast normalise over overlapping spatial cells
- Collect HOGs over detection window
- Feature vector \( f = [\ldots, \ldots, \ldots] \)

HoG Feature Extraction: Cells

- Tile window into 8 x 8 pixel cells
- Each cell represented by HOG
- Each cell contains a histogram of gradient orientations, weighted by gradient magnitude
HoG Feature Extraction: Blocks

2x2 block of cells

normalize [ , , , , ... , ]

“Each scalar cell response contributes several components to the final descriptor vector, each normalized with respect to a different block. This may seem redundant but good normalization is critical and including overlap significantly improves the performance.” Dalal&Triggs CVPR'05

HoG Design Choices

Parameters
- Gradient scale
- Orientation bins
- Block overlap area

Other choices
- RGB or Lab, Color/gray
- Block normalization
  \[ L^2 \text{-hys}, v \leftarrow \sqrt{\frac{1}{n} + \epsilon} \]
  or
  \[ L^1 \text{-sqrt}, v \leftarrow \sqrt{\frac{1}{n} + \epsilon} \]

Parameter / design choices were guided by extensive experimentation to determine empirical effects on detector performance (e.g. miss rate)

Dalal&Triggs Detector

- Default detector configuration:
  - RGB colour space with no gamma correction
  - [−1, 0, 1] gradient filter with no smoothing
  - linear gradient voting into 9 orientation bins in 0—180°
  - 16x16 pixel blocks of four 8x8 pixel cells
  - Gaussian spatial window with \( \sigma = 8 \) pixel
  - \( L^2 \)-Hys (Lowe-style clipped \( L^2 \) norm) block normalization
  - block spacing stride of 8 pixels (hence 4-fold coverage of each cell)
  - 64x128 detection window
  - linear SVM classifier.

Detector Architecture

Learning Phase
- Create normalised training data set
- Encode images into feature vectors
- Learn binary classifier
- Object/Non-object decision

Detection Phase
- Scan image at all scales and locations
- Run classifier to obtain object/non-object decisions
- Fuse multiple detections in 3-D position & scale space
- Object detections with bounding boxes

Positive and negative examples

+ thousands more...

+ millions more...
Evaluation Data Sets

<table>
<thead>
<tr>
<th>MIT pedestrian database</th>
<th>INRIA person database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Train</td>
</tr>
<tr>
<td>507 positive windows</td>
<td>1206 positive windows</td>
</tr>
<tr>
<td>Negative data unavailable</td>
<td>1218 negative images</td>
</tr>
<tr>
<td>200 positive windows</td>
<td>569 positive windows</td>
</tr>
<tr>
<td>Negative data unavailable</td>
<td>453 negative images</td>
</tr>
<tr>
<td>Overall 709 annotations+ reflections</td>
<td>Overall 1774 annotations+ reflections</td>
</tr>
</tbody>
</table>

Person detection with HoG & linear SVM

Soft (C=0.01) linear SVM trained with SVMLight.

[Dalal and Triggs, CVPR 2005]

To detect people at all locations and scales:

- Sliding window using learnt HOG template
- Post-processing using non-maxima suppression

To detect people at all locations and scales:

- Sliding window using learnt HOG template
- Post-processing using non-maxima suppression
To detect people at all locations and scales:

• Sliding window using learnt HOG template
• Post-processing using non-maxima suppression

To detect people at all locations and scales:

• Sliding window using learnt HOG template
• Post-processing using non-maxima suppression

Non-maximum Suppression across Scales

- Multi-scale dense scan of detection window
- Clip Detection Score
- Threshold
- Final detections

Apply robust mode detection, like mean shift

Overall Performance

MIT pedestrian database
INRIA person database

R/C-HOG give near perfect separation on MIT database
Have 1-2 order lower false positives than other descriptors
Dalal and Triggs Summary

- HoG feature representation
- Linear SVM classifier; sliding window detector
- Non-maximum suppression across scale
- Use of detector performance metrics to guide tuning of system parameters
- Detection rate 90% at $10^{-4}$ FP per window
- Slower than Viola-Jones detector