Lecture 2: The SVM classifier

• Review of linear classifiers
  • Linear separability
  • Perceptron

• Support Vector Machine (SVM) classifier
  • Wide margin
  • Cost function
  • Slack variables
  • Loss functions revisited
  • Optimization
Binary Classification

Given training data \((x_i, y_i)\) for \(i = 1 \ldots N\), with \(x_i \in \mathbb{R}^d\) and \(y_i \in \{-1, 1\}\), learn a classifier \(f(x)\) such that

\[
f(x_i) \begin{cases} 
  \geq 0 & y_i = +1 \\
  < 0 & y_i = -1
\end{cases}
\]

i.e. \(y_i f(x_i) > 0\) for a correct classification.
Linear separability

linearly separable

not linearly separable
A linear classifier has the form

\[ f(x) = \mathbf{w}^\top \mathbf{x} + b \]

- in 2D the discriminant is a line
- \( \mathbf{w} \) is the normal to the line, and \( b \) the bias
- \( \mathbf{w} \) is known as the weight vector
Linear classifiers

A linear classifier has the form

\[ f(x) = w^\top x + b \]

- in 3D the discriminant is a plane, and in nD it is a hyperplane

For a K-NN classifier it was necessary to `carry' the training data

For a linear classifier, the training data is used to learn \( w \) and then discarded

Only \( w \) is needed for classifying new data
The Perceptron Classifier

Given linearly separable data $x_i$ labelled into two categories $y_i = \{-1, 1\}$, find a weight vector $w$ such that the discriminant function

$$f(x_i) = w^\top x_i + b$$

separates the categories for $i = 1, \ldots, N$

• how can we find this separating hyperplane?

The Perceptron Algorithm

Write classifier as $f(x_i) = \tilde{w}^\top \tilde{x}_i + w_0 = w^\top x_i$

where $w = (\tilde{w}, w_0), x_i = (\tilde{x}_i, 1)$

• Initialize $w = 0$

• Cycle though the data points $\{x_i, y_i\}$

• if $x_i$ is misclassified then $w \leftarrow w + \alpha \text{sign}(f(x_i)) x_i$

• Until all the data is correctly classified
For example in 2D

- Initialize \( w = 0 \)
- Cycle though the data points \( \{ x_i, y_i \} \)
  - if \( x_i \) is misclassified then \( w \leftarrow w + \alpha \text{sign}(f(x_i))x_i \)
- Until all the data is correctly classified

\[ w \leftarrow w - \alpha x_i \]

NB after convergence \( w = \sum_{i=1}^{N} \alpha_i x_i \)
• if the data is linearly separable, then the algorithm will converge
• convergence can be slow …
• separating line close to training data
• we would prefer a larger margin for generalization
What is the best $w$?

- **maximum margin** solution: most stable under perturbations of the inputs
Support Vector Machine

\[ f(x) = \sum_{i} \alpha_i y_i (x_i^T x) + b \]

where \( f(x) \) is the decision function, \( \alpha_i \) are the Lagrange multipliers, \( y_i \) are the class labels, \( x_i \) are the support vectors, and \( b \) is the bias term.

Linearly separable data:

\[ w^T x + b = 0 \]
SVM – sketch derivation

• Since $\mathbf{w}^\top \mathbf{x} + b = 0$ and $c(\mathbf{w}^\top \mathbf{x} + b) = 0$ define the same plane, we have the freedom to choose the normalization of $\mathbf{w}$

• Choose normalization such that $\mathbf{w}^\top \mathbf{x}_+ + b = +1$ and $\mathbf{w}^\top \mathbf{x}_- + b = -1$ for the positive and negative support vectors respectively

• Then the margin is given by

$$\frac{\mathbf{w}}{||\mathbf{w}||} \cdot (\mathbf{x}_+ - \mathbf{x}_-) = \frac{\mathbf{w}^\top (\mathbf{x}_+ - \mathbf{x}_-)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$
Support Vector Machine

Margin = \frac{2}{||w||}

linearly separable data

\begin{align*}
  w^T x + b &= 1 \\
  w^T x &= 0 \\
  w^T x + b &= -1
\end{align*}
SVM – Optimization

• Learning the SVM can be formulated as an optimization:

\[
\max_w \frac{2}{\|w\|} \quad \text{subject to} \quad w^T x_i + b \begin{cases} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{cases} \text{ for } i = 1 \ldots N
\]

• Or equivalently

\[
\min_w \|w\|^2 \quad \text{subject to} \quad y_i \left( w^T x_i + b \right) \geq 1 \text{ for } i = 1 \ldots N
\]

• This is a quadratic optimization problem subject to linear constraints and there is a unique minimum
Linear separability again: What is the best \( w \)?

- the points can be linearly separated but there is a very narrow margin

- but possibly the large margin solution is better, even though one constraint is violated

In general there is a trade off between the margin and the number of mistakes on the training data
Introduce “slack” variables

\[ \xi_i \geq 0 \]

- for \( 0 < \xi \leq 1 \) point is between margin and correct side of hyperplane. This is a margin violation
- for \( \xi > 1 \) point is misclassified

\[ w^T x + b = 1 \]
\[ w^T x + b = -1 \]
\[ \text{Margin} = \frac{2}{||w||} \]

\[ \frac{\xi_i}{||w||} > \frac{2}{||w||} \]

\[ \frac{\xi_i}{||w||} < \frac{1}{||w||} \]

Misclassified point
Support Vector
Support Vector
"Soft" margin solution

The optimization problem becomes

\[
\min_{\mathbf{w} \in \mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||\mathbf{w}||^2 + C \sum_{i=1}^{N} \xi_i
\]

subject to

\[
y_i \left( \mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N
\]

• Every constraint can be satisfied if \( \xi_i \) is sufficiently large

• \( C \) is a regularization parameter:
  
  – small \( C \) allows constraints to be easily ignored \( \rightarrow \) large margin
  
  – large \( C \) makes constraints hard to ignore \( \rightarrow \) narrow margin
  
  – \( C = \infty \) enforces all constraints: hard margin

• This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, \( C \).
• data is linearly separable
• but only with a narrow margin
$C = \text{Infinity}$ hard margin
$C = 10$  soft margin
Application: Pedestrian detection in Computer Vision

Objective: detect (localize) standing humans in an image
- cf face detection with a sliding window classifier

Method: the HOG detector

- reduces object detection to binary classification
- does an image window contain a person or not?
Training data and features

• Positive data – 1208 positive window examples

• Negative data – 1218 negative window examples (initially)
Feature: histogram of oriented gradients (HOG)

- tile window into 8 x 8 pixel cells
- each cell represented by HOG

Feature vector dimension = 16 x 8 (for tiling) x 8 (orientations) = 1024
Averaged positive examples
Training (Learning)

- Represent each example window by a HOG feature vector

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

- Train a SVM classifier

Testing (Detection)

- Sliding window classifier

\[ f(x) = \mathbf{w}^\top \mathbf{x} + b \]
Dalal and Triggs, CVPR 2005
Learned model

\[ f(x) = w^\top x + b \]
What do negative weights mean?

\[ w_x > 0 \]
\[ (w_+ - w_-)x > 0 \]
\[ w_+ > w_-x \]

pedestrian model > pedestrian background model

Complete system should compete pedestrian/pillar/doorway models

Discriminative models come equipped with own bg
(avoid firing on doorways by penalizing vertical edges)

Slide from Deva Ramanan
Optimization

Learning an SVM has been formulated as a constrained optimization problem over \( w \) and \( \xi \)

\[
\min_{w \in \mathbb{R}^d, \xi_i \in \mathbb{R}^+} \|w\|^2 + C \sum_{i}^{N} \xi_i \text{ subject to } y_i (w^T x_i + b) \geq 1 - \xi_i \text{ for } i = 1 \ldots N
\]

The constraint \( y_i (w^T x_i + b) \geq 1 - \xi_i \), can be written more concisely as

\[
y_i f(x_i) \geq 1 - \xi_i
\]

which, together with \( \xi_i \geq 0 \), is equivalent to

\[
\xi_i = \max(0, 1 - y_i f(x_i))
\]

Hence the learning problem is equivalent to the unconstrained optimization problem over \( w \)

\[
\min_{w \in \mathbb{R}^d} \|w\|^2 + C \sum_{i}^{N} \max(0, 1 - y_i f(x_i))
\]

regularization \hspace{1cm} loss function
Loss function

\[
\min_{\mathbf{w} \in \mathbb{R}^d} \|\mathbf{w}\|^2 + C \sum_{i} \max(0, 1 - y_if(x_i))
\]

Points are in three categories:

1. \(y_if(x_i) > 1\)
   - Point is outside margin.
   - No contribution to loss

2. \(y_if(x_i) = 1\)
   - Point is on margin.
   - No contribution to loss.
   - As in hard margin case.

3. \(y_if(x_i) < 1\)
   - Point violates margin constraint.
   - Contributes to loss
• SVM uses “hinge” loss  \( \max (0, 1 - y_if(x_i)) \)
• an approximation to the 0-1 loss
\[
\min_{\mathbf{w} \in \mathbb{R}^d} C \sum_{i}^{N} \max(0, 1 - y_i f(x_i)) + \|\mathbf{w}\|^2
\]

• Does this cost function have a unique solution?
• Does the solution depend on the starting point of an iterative optimization algorithm (such as gradient descent)?

If the cost function is **convex**, then a locally optimal point is globally optimal (provided the optimization is over a convex set, which it is in our case)
Convex functions

$D$ – a domain in $\mathbb{R}^n$.

A convex function $f : D \to \mathbb{R}$ is one that satisfies, for any $x_0$ and $x_1$ in $D$:

$$f((1 - \alpha)x_0 + \alpha x_1) \leq (1 - \alpha)f(x_0) + \alpha f(x_1).$$

Line joining $(x_0, f(x_0))$ and $(x_1, f(x_1))$ lies above the function graph.
Convex function examples

A non-negative sum of convex functions is convex.
SVM

\[
\min_{w \in \mathbb{R}^d} C \sum_{i}^{N} \max(0, 1 - y_i f(x_i)) + \|w\|^2
\]

convex
Gradient (or steepest) descent algorithm for SVM

To minimize a cost function $C(w)$ use the iterative update

$$ w_{t+1} \leftarrow w_t - \eta_t \nabla wC(w_t) $$

where $\eta$ is the learning rate.

First, rewrite the optimization problem as an average

$$ \min_w C(w) = \frac{\lambda}{2} ||w||^2 + \frac{1}{N} \sum_i \max(0, 1 - y_i f(x_i)) $$

$$ = \frac{1}{N} \sum_i \left( \frac{\lambda}{2} ||w||^2 + \max(0, 1 - y_i f(x_i)) \right) $$

(with $\lambda = 2/(NC)$ up to an overall scale of the problem) and $f(x) = w^T x + b$

Because the hinge loss is not differentiable, a sub-gradient is computed
Sub-gradient for hinge loss

\[ L(x_i, y_i; w) = \max(0, 1 - y_i f(x_i)) \quad f(x_i) = w^\top x_i + b \]
Sub-gradient descent algorithm for SVM

\[ C(w) = \frac{1}{N} \sum_{i}^{N} \left( \frac{\lambda}{2} ||w||^2 + \mathcal{L}(x_i, y_i; w) \right) \]

The iterative update is

\[
w_{t+1} \leftarrow w_t - \eta \nabla w_t C(w_t)
\]

\[
\leftarrow w_t - \eta \frac{1}{N} \sum_{i}^{N} (\lambda w_t + \nabla w \mathcal{L}(x_i, y_i; w_t))
\]

where \( \eta \) is the learning rate.

Then each iteration \( t \) involves cycling through the training data with the updates:

\[
w_{t+1} \leftarrow w_t - \eta (\lambda w_t - y_i x_i) \quad \text{if} \ y_i f(x_i) < 1
\]

\[
\leftarrow w_t - \eta \lambda w_t \quad \text{otherwise}
\]

In the Pegasos algorithm the learning rate is set at \( \eta_t = \frac{1}{\lambda t} \)
Pegasos – Stochastic Gradient Descent Algorithm

Randomly sample from the training data

![Graph showing energy vs iteration]

![Graph showing data distribution]
Background reading and more …

- Next lecture – see that the SVM can be expressed as a sum over the support vectors:

$$f(x) = \sum_{i} \alpha_{i} y_{i} (x_{i}^\top x) + b$$

- On web page:
  
  http://www.robots.ox.ac.uk/~az/lectures/ml

- links to SVM tutorials and video lectures

- MATLAB SVM demo