Chapter 2

A Brief Introduction to Support Vector Machine (SVM)

January 25, 2011
Overview

• A new, powerful method for 2-class classification
  – Original idea: Vapnik, 1965 for linear classifiers
  – SVM, Cortes and Vapnik, 1995
  – Became very hot since 2001

• Better generalization (less overfitting)

• Can do linearly unseparable classification with global optimal

• Key ideas
  – Use kernel function to transform low dimensional training samples to higher dim (for linear separability problem)
  – Use quadratic programming (QP) to find the best classifier boundary hyperplane (for global optima)
Linear Classifiers

\[ f(x, w, b) = \text{sign}(w \cdot x - b) \]

- denotes +1
- denotes -1

How would you classify this data?
Linear Classifiers

\[ f(x, w, b) = \text{sign}(w \cdot x - b) \]

- \( \alpha \)
- \( x \rightarrow f \rightarrow y^{\text{est}} \)

- denotes +1
- denotes -1

How would you classify this data?
Linear Classifiers

\[ f(x, w, b) = \text{sign}(w \cdot x - b) \]

• denotes +1
• denotes -1

How would you classify this data?
Linear Classifiers

\[ f(x, w, b) = \text{sign}(w \cdot x - b) \]

- denotes +1
- denotes -1

How would you classify this data?
Linear Classifiers

\[
f(x, w, b) = \text{sign}(w \cdot x - b)
\]

- \(f\) denotes +1
- \(f\) denotes -1

Any of these would be fine..

..but which is best?

Support Vector Machines: Slide 7
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

\[ f(x, w, b) = \text{sign}(w \cdot x - b) \]
The maximum margin linear classifier is the linear classifier with maximum margin.

This is the simplest kind of SVM (Called an LSVM)
The maximum margin linear classifier is the linear classifier with maximum margin. This is the simplest kind of SVM (Called an LSVM)

\[ f(x, w, b) = \text{sign}(w \cdot x - b) \]

Support Vectors are those datapoints that the margin pushes up against

- denotes +1
- denotes -1

Support Vector Machines: Slide 10
Why Maximum Margin?

1. Intuitively this feels safest.
2. If we’ve made a small error in the location of the boundary this gives us least chance of causing a misclassification.
3. It also helps generalization.
4. There’s some theory that this is a good thing.
5. Empirically it works very very well.

Support Vectors are those datapoints that the margin pushes up against.

This is the simplest kind of SVM (Called an LSVM)

Support Vector Machines: Slide 11
Computing the margin width

How do we compute $M$ in terms of $w$ and $b$?

- **Plus-plane** = \( \{ \mathbf{x} : w \cdot \mathbf{x} + b = +1 \} \)
- **Minus-plane** = \( \{ \mathbf{x} : w \cdot \mathbf{x} + b = -1 \} \)
- \( M = \frac{2}{\sqrt{w \cdot w}} \)
Learning the Maximum Margin Classifier

Given a guess of $w$ and $b$ we can

- Compute whether all data points in the correct half-planes
- Compute the width of the margin

So now we just need to write a program to search the space of $w$’s and $b$’s to find the widest margin that matches all the datapoints. *How?*

Gradient descent? Simulated Annealing?

\[
M = \text{Margin Width} = \frac{2}{\sqrt{w \cdot w}}
\]
Learning via Quadratic Programming

- QP is a well-studied class of optimization algorithms to maximize a quadratic function of some real-valued variables subject to linear constraints.
- Minimize both $\mathbf{w} \cdot \mathbf{w}$ (to maximize $M$) and misclassification error.
Quadratic Programming

Find $\arg \max_u \ c + d^T u + \frac{u^T R u}{2}$

Quadratic criterion

Subject to
\[
\begin{align*}
\sum a_{1i} u_i & \leq b_1 \\
\sum a_{2i} u_i & \leq b_2 \\
& \vdots \\
\sum a_{ni} u_i & \leq b_n
\end{align*}
\]
\text{ } \quad n \text{ additional linear inequality constraints}

And subject to
\[
\begin{align*}
\sum a_{(n+1)i} u_i & = b_{(n+1)} \\
\sum a_{(n+2)i} u_i & = b_{(n+2)} \\
& \vdots \\
\sum a_{(n+e)i} u_i & = b_{(n+e)}
\end{align*}
\]
\text{ } \quad n \text{ additional linear equality/equality constraints}

Support Vector Machines: Slide 15
Learning Maximum Margin with Noise

Given guess of $w$, $b$ we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

Assume $R$ datapoints, each $(x_k, y_k)$ where $y_k = +/- 1$

What should our quadratic optimization criterion be? How many constraints will we have? What should they be?
Learning Maximum Margin with Noise

Given guess of $\mathbf{w}, b$ we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

Assume $R$ datapoints, each $(\mathbf{x}_k, y_k)$ where $y_k = +/- 1$

What should our quadratic optimization criterion be?

Minimize

$$
\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^{R} \varepsilon_k
$$

$\varepsilon_k = \text{distance of error points to their correct place}$

How many constraints will we have? **2R**

What should they be?

$$
\mathbf{w} \cdot \mathbf{x}_k + b \geq 1 - \varepsilon_k \text{ if } y_k = 1
$$

$$
\mathbf{w} \cdot \mathbf{x}_k + b \leq -1 + \varepsilon_k \text{ if } y_k = -1
$$

$\varepsilon_k \geq 0 \text{ for all } k$
From LSVM to general SVM

Suppose we are in 1-dimension

What would SVMs do with this data?
Not a big surprise

Support Vector Machines: Slide 19
Harder 1-dimensional dataset

Points are not linearly separable.

What can we do now?

$\text{x}=0$

Support Vector Machines: Slide 20
Harder 1-dimensional dataset

Transform the data points from 1-dim to 2-dim by some nonlinear basis function (called **Kernel functions**)

These points sometimes are called **feature vectors**.

\[ \mathbf{z}_k = (x_k, x_k^2) \]
Harder 1-dimensional dataset

These points are linearly separable now!

Boundary can be found by QP

$z_k = (x_k, x_k^2)$
Common SVM basis functions

\[ z_k = \text{(polynomial terms of } x_k \text{ of degree 1 to } q \text{ )} \]

\[ z_k = \text{(radial basis functions of } x_k \text{ )} \]

\[ z_k[j] = \varphi_j(x_k) = \text{KernelFn}\left(\frac{|x_k - c_j|}{KW}\right) \]

\[ z_k = \text{(sigmoid functions of } x_k \text{ )} \]
Doing multi-class classification

- SVMs can only handle two-class outputs (i.e., a categorical output variable with variety 2).
- What can be done?
- Answer: with N classes, learn N SVM’s
  - SVM 1 learns “Output==1” vs “Output != 1”
  - SVM 2 learns “Output==2” vs “Output != 2”
  - ...
  - SVM N learns “Output==N” vs “Output != N”
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.
Compare SVM with NN

- **Similarity**
  - SVM + sigmoid kernel ~ two-layer feedforward NN
  - SVM + Gaussian kernel ~ RBF network
  - For most problems, SVM and NN have similar performance

- **Advantages**
  - Based on sound mathematics theory
  - Learning result is more robust
  - Over-fitting is not common
  - Not trapped in local minima (because of QP)
  - Fewer parameters to consider (kernel, error cost \( C \))
  - Works well with fewer training samples (number of support vectors do not matter much)

- **Disadvantages**
  - Problem need to be formulated as 2-class classification
  - Learning takes long time (QP optimization)
Advantages and Disadvantages of SVM

- **Advantages**
  - prediction accuracy is generally high
  - robust, works when training examples contain errors
  - fast evaluation of the learned target function

- **Criticism**
  - long training time
  - difficult to understand the learned function (weights)
  - not easy to incorporate domain knowledge
SVM Related Links

- Representative implementations
  - LIBSVM: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - SVM-light: simpler but performance is not better than LIBSVM, support only binary classification and only C language
  - SVM-torch: another recent implementation also written in C.