Support Vector Machines

Modified from Prof. Andrew W. Moore's Lecture Notes

www.cs.cmu.edu/~awm

History

• SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis
• SVMs introduced by Boser, Guyon, Vapnik in COLT-92
• Now an important and active field of all Machine Learning research.

Linear Classifiers

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

- \( w \) denotes +1
- \( b \) denotes -1

How would you classify this data?

e.g. \( x = (x_1, x_2) \), \( w = (w_1, w_2) \), \( w \cdot x = x_1w_1 + w_2x_2 \)
\[ \text{sign}(w \cdot x + b) = +1 \iff x_1w_1 + w_2x_2 - b > 0 \]
Linear Classifiers

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

Denotes +1
Denotes -1

Any of these would be fine...

...but which is best?

e.g. \( x = (x_1, x_2) \), \( w = (w_1, w_2) \), \( w \cdot x = w_1 x_1 + w_2 x_2 \)
\[ \text{sign}(w \cdot x + b) = +1 \iff x_1 w_1 + w_2 x_2 + b > 0 \]

Classifier Margin

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

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

e.g. \( x = (x_1, x_2) \), \( w = (w_1, w_2) \), \( w \cdot x = x_1 w_1 + w_2 x_2 \)
\[ \text{sign}(w \cdot x + b) = +1 \iff x_1 w_1 + w_2 x_2 - b > 0 \]

Maximum Margin

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

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

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

Support Vectors are those datapoints that the margin pushes up against

Why Maximum Margin?

1. Intuitively this feels safest.
2. Empirically it works very well.
3. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
4. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.

Estimate the Margin

\[ d(x) = \frac{||x - w'||}{\sqrt{||w'||^2 + \sum_{i=1}^{d} n_i^2}} \]

X – Vector
W – Normal Vector
b – Scale Value

http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html
Estimate the Margin

- What is the expression for margin, given $w$ and $b$?

$\text{margin} = \arg\min_{D} d(x) = \arg\min_{x\in D} \frac{|x \cdot w + b|}{\sum_{i=1}^{d} w_i^2}$

Maximize Margin

$\text{argmax}_{D} \text{margin}(w, b, D)$

$\Rightarrow \arg\max_{w, b} \arg\min_{x\in D} \frac{|x \cdot w + b|}{\sum_{i=1}^{d} w_i^2}$

Maximize Margin

$\text{Strategy:}$

$\forall x_i \in D : y_i(x_i \cdot w + b) \geq 0$

$\Rightarrow \arg\max_{w, b} \sum_{i=1}^{d} w_i^2$

We have

$\arg\max_{w, b} \frac{|x \cdot w + b|}{\sum_{i=1}^{d} w_i^2} = \arg\max_{w, b} \frac{1}{\sum_{i=1}^{d} w_i^2}$

Thus,

$\Rightarrow \arg\max_{w, b} \frac{|x \cdot w + b|}{\sum_{i=1}^{d} w_i^2} = \arg\max_{w, b} \frac{1}{\sum_{i=1}^{d} w_i^2} = \arg\min_{w, b} \sum_{i=1}^{d} w_i^2$

Maximum Margin Linear Classifier

$\Rightarrow \{w^*, b^*\} = \arg\max_{w, b} \sum_{i=1}^{d} w_i^2$

subject to

$\forall y_i (\frac{r^T}{r} x_i + b) \geq 1$

$\Rightarrow \sum_{i=1}^{N} w_i^2$

How does it come?

$\Rightarrow \{w^*, b^*\} = \arg\max_{w, b} \sum_{i=1}^{d} w_i^2$

subject to

$\forall x_i \in D : y_i(x_i \cdot w + b) \geq 0$

We have

$\arg\max_{w} \frac{|x \cdot w + b|}{\sum_{i=1}^{d} w_i^2} = \arg\min_{w} \frac{1}{\sum_{i=1}^{d} w_i^2}$

Thus,

$\Rightarrow \arg\max_{w} \frac{|x \cdot w + b|}{\sum_{i=1}^{d} w_i^2} = \arg\max_{w} \frac{1}{\sum_{i=1}^{d} w_i^2} = \arg\min_{w} \sum_{i=1}^{d} w_i^2$

Maximum Margin Linear Classifier

$\Rightarrow \{w^*, b^*\} = \arg\max_{w, b} \sum_{i=1}^{d} w_i^2$

subject to

$\forall y_i (\frac{r^T}{r} x_i + b) \geq 1$

$\Rightarrow \sum_{i=1}^{N} w_i^2$

• How to solve it?
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.
• Detail solution of Quadratic Programming
  • Convex Optimization Stephen P. Boyd
  • Online Edition, Free for Download

www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

Quadratic Programming

\[
\begin{align*}
\text{Find } & \quad \arg \max_{\mathbf{u}} \quad c + \mathbf{d}^T \mathbf{u} + \frac{\mathbf{u}^T \mathbf{R} \mathbf{u}}{2} \\
\text{Subject to } & \quad a_1 u_1 + a_2 u_2 + \ldots + a_n u_n \leq b_1 \\
& \quad a_2 u_2 + a_3 u_3 + \ldots + a_n u_n \leq b_2 \\
& \quad \vdots \\
& \quad a_n u_1 + a_1 u_2 + \ldots + a_n u_n \leq b_n
\end{align*}
\]

n additional linear inequality constraints

Quadratic Programming for the Linear Classifier

\[
\begin{align*}
\{ \mathbf{w}^*, b^* \} = & \min_{\mathbf{w}, b} \sum_i n_i^2 \\
\text{subject to } & \quad y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for all training data } (\mathbf{x}_i, y_i)
\end{align*}
\]

\[
\{ \mathbf{w}^*, b^* \} = \arg \max_{\mathbf{w}, b} \left\{ 0 : \mathbf{w} \cdot \mathbf{x}_i + b \right\}
\]

\[
\begin{align*}
& \quad y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \\
& \quad y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ inequality constraints}
\end{align*}
\]

• Popular Tools - LibSVM

Uh-oh! 
This is going to be a problem!
What should we do?

Idea 1:
Find minimum \( \mathbf{w}, \mathbf{w} \), while minimizing number of training set errors.

Problem: Two things to minimize makes for an ill-defined optimization

Uh-oh! 
This is going to be a problem!
What should we do?

Idea 1:
Find minimum \( \mathbf{w}, \mathbf{w} \), while minimizing number of training set errors.

Problem: Two things to minimize makes for an ill-defined optimization
Uh-oh!

This is going to be a problem!
What should we do?

Idea 1.1:
Minimize
\[ w^T w + C \text{ (#train errors)} \]

There’s a serious practical problem that’s about to make us reject this approach. Can you guess what it is?

Support Vector Machine (SVM) for Noisy Data

\[
\begin{align*}
\{w^*, b^*\} &= \text{argmin}_{w,b} \sum_{i=1}^{d} d_i^2 + C \sum_{j=1}^{N} \varepsilon_j \\
y_1 \left( w \cdot x_1 + b \right) &\geq 1 - \varepsilon_1, \varepsilon_1 \geq 0 \\
y_2 \left( w \cdot x_2 + b \right) &\geq 1 - \varepsilon_2, \varepsilon_2 \geq 0 \\
\vdots \\
y_N \left( w \cdot x_N + b \right) &\geq 1 - \varepsilon_N, \varepsilon_N \geq 0
\end{align*}
\]

- Balance the tradeoff between margin and classification errors

How do we determine the appropriate value for \( C \)?
The Dual Form of QP

Maximize \( \sum_{k=1}^{n} a_k \)\(-\frac{1}{2} \sum_{k=1}^{n} \sum_{l=1}^{n} a_k a_l Q_{kl} \) where \( Q_{kl} = y_k y_l (x_k, x_l) \)

Subject to these constraints: \( 0 \leq a_k \leq C \) \( \forall k \sum_{k=1}^{n} a_k y_k = 0 \)

Then define:

\[ w = \sum_{k=1}^{n} a_k y_k x_k \]

Support Vectors

\[ w \cdot \begin{pmatrix} x \end{pmatrix} + b = -1 \]

\( w \cdot \begin{pmatrix} x \end{pmatrix} + b = 1 \)

Decision boundary is determined only by those support vectors!

An Equivalent QP

Maximize \( \sum_{k=1}^{n} a_k \)\(-\frac{1}{2} \sum_{k=1}^{n} \sum_{l=1}^{n} a_k a_l Q_{kl} \) where \( Q_{kl} = y_k y_l (x_k, x_l) \)

Subject to these constraints: \( 0 \leq a_k \leq C \) \( \forall k \sum_{k=1}^{n} a_k y_k = 0 \)

Then define:

\[ w = \sum_{k=1}^{n} a_k y_k x_k \]

Detectors with \( a_k > 0 \) will be the support vectors.

Then classify with:

\( f(\begin{pmatrix} x \end{pmatrix}, w, b) = \text{sign}(w \cdot \begin{pmatrix} x \end{pmatrix} + b) \)

How to determine \( b \)?

An Equivalent QP: Determine \( b \)

\[ \begin{bmatrix} 1 & w^T \end{bmatrix} = \arg \min_{b \in \mathbb{R}} \sum_{k=1}^{n} y_k \left( \begin{array}{c} 1 \end{array} \right) - \left( \begin{array}{c} 1 \end{array} \right) y_k \cdot \left( \begin{array}{c} b \end{array} \right) \]

Fix \( w \)

\[ \begin{array}{c} y_1 (\begin{pmatrix} w \cdot \begin{pmatrix} x \end{pmatrix} + b \end{pmatrix}) \geq 1 - e_1 \varepsilon_1 \geq 0 \\
\vdots \\
\end{array} \]

A linear programming problem!

Parameter \( c \) is used to control the fitness
Feature Transformation?
- The problem is non-linear
- Find some trick to transform the input
- Linear separable after Feature Transformation
- What Features should we use?

Basic Idea:

Suppose we’re in 1-dimension

What would SVMs do with this data?

Suppose we’re in 1-dimension

Not a big surprise

Harder 1-dimensional dataset

That’s wiped the smirk off SVM’s face.

What can be done about this?

Harder 1-dimensional dataset

Map the data from low-dimensional space to high-dimensional space.
Let’s permit them here too.

Feature Enumeration

Z_k = (x_k, x_k^2)
Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:

\[ \Phi: \mathbb{R}^d \rightarrow \mathbb{R}^D \]

\[ (x_1, x_2) \rightarrow (x_1, x_2, x_1^2, x_1 x_2, x_2^2) \]

- Polynomial features for the XOR problem

Efficiency Problem in Computing Feature

- Feature space Mapping

\[ \Phi: X \rightarrow \mathcal{H} \]

where \( \mathcal{H} \) is a dot product space and \( \Phi \) maps the mapping from \( X \) to \( \mathcal{H} \).

- Example: all 2 degree Monomials

\[ (x_1, x_2) \rightarrow (x_1, x_2, x_1 x_2, x_1^2, x_2^2) \]

This use of kernel function to avoid carrying out \( \Phi(x) \) explicitly is known as the kernel trick.

\[ \langle \Phi(x_1), \Phi(x_2) \rangle = (x_1, x_2, x_1 x_2, x_1^2, x_2^2) \cdot (x_1, x_2, x_1 x_2, x_1^2, x_2^2) \]

\[ =: k(x, x') \]

\[ \rightarrow \text{the dot product in } \mathcal{H} \text{ can be computed in } \mathbb{R}^2 \]

- Could solve complicated Non-Linear Problems
- \( y \) and \( c \) control the complexity of decision boundary

\[ y = \frac{1}{\sigma^2} \]

Common SVM basis functions

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

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

\[ z_k = \exp(-\frac{|x_1 - x_2|^2}{\sigma^2}) \]

- \( z_k = \text{ (sigmoid functions of } x_k \text{) } \)

"Radius basis functions" for the XOR problem
How to Control the Complexity
Which reasoning below is the most probable?

- Bob got up and found that breakfast was ready
  - His Child (Underfitting)
  - His Wife (Reasonable)
  - The Alien (Overfitting)

How to Control the Complexity

- SVM is powerful to approximate any training data
- The complexity affects the performance on new data
- SVM supports parameters for controlling the complexity
- SVM does not tell you how to set these parameters
- Determine the Parameters by Cross-Validation

SVM Performance

- Anecdotally they work very very well indeed.
- Example: They are currently the best-known classifier on a well-studied hand-written-character recognition benchmark
- There is a lot of excitement and religious fervor about SVMs.
- Despite this, some practitioners are a little skeptical.

References

- An excellent tutorial on VC-dimension and Support Vector Machines:
- The VC/SRM/SVM Bible: (Not for beginners including myself)
  - SMO in Weka