10-701
Machine Learning
Support Vector Machine
Types of classifiers

• We can divide the large variety of classification approaches into roughly three major types

  1. Instance based classifiers
     - Use observation directly (no models)
     - e.g. K nearest neighbors

  2. Generative:
     - build a generative statistical model
     - e.g., Bayesian networks

  3. Discriminative
     - directly estimate a decision rule/boundary
     - e.g., decision tree
Recall our regression classifiers

\[ +1 \text{ if } \text{sign}(w^T x + b) \geq 0 \]

\[ -1 \text{ if } \text{sign}(w^T x + b) < 0 \]
Regression classifiers

Recall our regression classifiers

Line closer to the blue nodes since many of them are far away from the boundary
Regression classifiers

Recall our regression classifiers

\[
\min_w \sum_i (y_i - w^T x_i)^2
\]

Goes over all points \(x\) (even for logistic regression)

Line closer to the blue nodes since many of them are far away from the boundary
Regression classifiers

Recall our regression classifiers

Many more possible classifiers

\[ \min_w \sum_i (y_i - w^T x_i)^2 \]

Goes over all points \( x \) (even in LR settings)

Line closer to the blue nodes since many of them are far away from the boundary
Max margin classifiers

- Instead of fitting all points, focus on boundary points.
- Learn a boundary that leads to the largest margin from both sets of points (that is, largest distance to the closest point on either side).

From all the possible boundary lines, this leads to the largest margin on both sides.
Max margin classifiers

- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest margin from points on both sides

Why?
- Intuitive, ‘makes sense’
- Some theoretical support
- Works well in practice
Max margin classifiers

- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest margin from points on both sides

Also known as linear support vector machines (SVMs)

These are the vectors supporting the boundary
Specifying a max margin classifier

Predict class +1

\[ w^T x + b = +1 \]
\[ w^T x + b = 0 \]
\[ w^T x + b = -1 \]

Predict class -1

Class +1 plane
boundary
Class -1 plane

Classify as +1 if \( w^T x + b \geq 1 \)

Classify as -1 if \( w^T x + b \leq -1 \)

Undefined if \(-1 < w^T x + b < 1 \)
Specifying a max margin classifier

Predict class +1

\( w^T x + b = +1 \)
\( w^T x + b = 0 \)
\( w^T x + b = -1 \)

Predict class -1

Classify as +1 if \( w^T x + b \geq 1 \)

Classify as -1 if \( w^T x + b \leq -1 \)

Undefined if \( -1 < w^T x + b < 1 \)

Is the linear separation assumption realistic?

We will deal with this shortly, but let's assume it for now.
Maximizing the margin

Classify as +1 if \( w^T x + b \geq 1 \)
Classify as -1 if \( w^T x + b \leq -1 \)
Undefined if \(-1 < w^T x + b < 1\)

• Let's define the width of the margin by \( M \)
• How can we encode our goal of maximizing \( M \) in terms of our parameters (\( w \) and \( b \))?
• Let's start with a few observations
Maximizing the margin

Classify as +1 if $w^T x + b \geq 1$
Classify as -1 if $w^T x + b \leq -1$
Undefined if $-1 < w^T x + b < 1$

• Observation 1: the vector w is orthogonal to the +1 plane
• Why?

Let $u$ and $v$ be two points on the +1 plane, then for the vector defined by $u$ and $v$ we have $w^T(u-v) = 0$

Corollary: the vector w is orthogonal to the -1 plane
Maximizing the margin

Classify as +1 if \( w^T x + b \geq 1 \)
Classify as -1 if \( w^T x + b \leq -1 \)
Undefined if \(-1 < w^T x + b < 1\)

• Observation 1: the vector \( w \) is orthogonal to the +1 and -1 planes
• Observation 2: if \( x^+ \) is a point on the +1 plane and \( x^- \) is the closest point to \( x^+ \) on the -1 plane then

\[
x^+ = \lambda w + x^-
\]

Since \( w \) is orthogonal to both planes we need to ‘travel’ some distance along \( w \) to get from \( x^+ \) to \( x^- \)
Putting it together

We can now define $M$ in terms of $w$ and $b$

\[ w^T x^+ + b = +1 \]
\[ \implies \]
\[ w^T (\lambda w + x^-) + b = +1 \]
\[ \implies \]
\[ w^T x^- + b + \lambda w^T w = +1 \]
\[ \implies \]
\[ -1 + \lambda w^T w = +1 \]
\[ \implies \]
\[ \lambda = 2/w^T w \]
Putting it together

- $w^T x + b = +1$
- $w^T x + b = 0$
- $w^T x + b = -1$
- $x^+ = \lambda w + x^-$
- $|x^+ - x^-| = M$
- $\lambda = 2/w^T w$

We can now define $M$ in terms of $w$ and $b$.

$$M = |x^+ - x^-|$$

$$\Rightarrow$$

$$M = |\lambda w| = \lambda |w| = \lambda \sqrt{w^T w}$$

$$\Rightarrow$$

$$M = 2 \frac{\sqrt{w^T w}}{w^T w} = \frac{2}{\sqrt{w^T w}}$$
Finding the optimal parameters

We can now search for the optimal parameters by finding a solution that:

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

Several optimization methods can be used: Gradient descent, simulated annealing, EM etc.

$$M = \frac{2}{\sqrt{w^T w}}$$
Quadratic programming (QP)

Quadratic programming solves optimization problems of the following form:

\[ \min_U \frac{u^T Ru}{2} + d^T u + c \]

subject to \( n \) inequality constraints:

\[ a_{11}u_1 + a_{12}u_2 + \ldots \leq b_1 \]
\[ \vdots \]
\[ a_{n1}u_1 + a_{n2}u_2 + \ldots \leq b_n \]

and \( k \) equivalency constraints:

\[ a_{n+1,1}u_1 + a_{n+1,2}u_2 + \ldots = b_{n+1} \]
\[ \vdots \]
\[ a_{n+k,1}u_1 + a_{n+k,2}u_2 + \ldots = b_{n+k} \]

When a problem can be specified as a QP problem we can use solvers that are better than gradient descent or simulated annealing.

- \( u \) - vector (unknown)
- \( R \) – squared matrix
- \( d \) – vector
- \( c \) - scalar

Quadratic term
SVM as a QP problem

\[
\text{Min } (w^T w)/2 \\
\text{subject to the following inequality constraints:} \\
\text{For all } x \text{ in class } +1 \\
w^T x + b \geq 1 \\
\text{For all } x \text{ in class } -1 \\
w^T x + b \leq -1 \\
\]

A total of n constraints if we have n input samples

\[
M = \frac{2}{\sqrt{w^T w}} \\
\]

\[
\min_U \frac{u^T Ru}{2} + d^T u + c \\
\text{subject to } n \text{ inequality constraints:} \\
a_{11}u_1 + a_{12}u_2 + \ldots \leq b_1 \\
\vdots \quad \vdots \quad \vdots \\
a_{n1}u_1 + a_{n2}u_2 + \ldots \leq b_n \\
\text{and } k \text{ equivalency constraints:} \\
a_{n+1,1}u_1 + a_{n+1,2}u_2 + \ldots = b_{n+1} \\
\vdots \quad \vdots \quad \vdots \\
a_{n+k,1}u_1 + a_{n+k,2}u_2 + \ldots = b_{n+k}
\]
Non linearly separable case

- So far we assumed that a linear plane can perfectly separate the points
- But this is not usually the case
  - noise, outliers

How can we convert this to a QP problem?
- Minimize training errors?
  \[
  \min w^T w \\
  \min \text{#errors}
  \]
- Penalize training errors:
  \[
  \min w^T w + C\times\text{#errors}
  \]

Hard to solve (two minimization problems)

Hard to encode in a QP problem
Non linearly separable case

• Instead of minimizing the number of misclassified points we can minimize the distance between these points and their correct plane.

The new optimization problem is:

\[
\min_w \frac{1}{2} w^T w + \sum_{i=1}^{n} C \varepsilon_i \\
\text{subject to the following inequality constraints:}
\]

For all \( x_i \) in class + 1

\[
w^T x + b \geq 1 - \varepsilon_i
\]

For all \( x_i \) in class - 1

\[
w^T x + b \leq -1 + \varepsilon_i
\]

These are also support vectors since they impact the parameters of the decision boundary.

Wait. Are we missing something?
Final optimization for non linearly separable case

The new optimization problem is:

$$
\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C \varepsilon_i
$$

subject to the following inequality constraints:

For all $x_i$ in class $+1$

$$
w^T x + b \geq 1 - \varepsilon_i
$$

For all $x_i$ in class $-1$

$$
w^T x + b \leq -1 + \varepsilon_i
$$

For all $i$

$$
\varepsilon_i \geq 0
$$

A total of $n$ constraints

Another $n$ constraints
Where we are

Two optimization problems: For the separable and non separable cases

\[
\min_w \frac{w^T w}{2}
\]

For all \( x \) in class + 1

\[ w^T x + b \geq 1 \]

For all \( x \) in class - 1

\[ w^T x + b \leq -1 \]

\[
\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C \varepsilon_i
\]

For all \( x_i \) in class + 1

\[ w^T x + b \geq 1 - \varepsilon_i \]

For all \( x_i \) in class - 1

\[ w^T x + b \leq -1 + \varepsilon_i \]

For all \( i \)

\[ \varepsilon_i \geq 0 \]
Where we are

Two optimization problems: For the separable and non separable cases

Min \( (w^T w) / 2 \)

For all \( x \) in class + 1
\( w^T x + b \geq 1 \)

For all \( x \) in class - 1
\( w^T x + b \leq -1 \)

\[
\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C \varepsilon_i
\]

For all \( x_i \) in class + 1
\( w^T x_i + b \geq 1 - \varepsilon_i \)

For all \( x_i \) in class - 1
\( w^T x_i + b \leq -1 + \varepsilon_i \)

For all \( i \)
\( \varepsilon_i \geq 0 \)

• Instead of solving these QPs directly we will solve a dual formulation of the SVM optimization problem

• The main reason for switching to this type of representation is that it would allow us to use a neat trick that will make our lives easier (and the run time faster)
An alternative (dual) representation of the SVM QP

• We will start with the linearly separable case
• Instead of encoding the correct classification rule and constraint we will use LaGrange multiplies to encode it as part of the our minimization problem

\[
\begin{align*}
\text{Min } & (w^T w)/2 \\
\text{For all } & x \text{ in class } +1 \\
w^T x + b & \geq 1 \\
\text{For all } & x \text{ in class } -1 \\
w^T x + b & \leq -1
\end{align*}
\]

Why?

\[
\begin{align*}
\text{Min } & (w^T w)/2 \\
(w^T x_i + b) y_i & \geq 1
\end{align*}
\]
An alternative (dual) representation of the SVM QP

- We will start with the linearly separable case
- Instead of encoding the correct classification rule as a constraint we will use Lagrange multiplies to encode it as part of the our minimization problem

Recall that Lagrange multipliers can be applied to turn the following problem:

\[
\begin{align*}
\min_x x^2 \\
s.t. \ x \geq b
\end{align*}
\]

To

\[
\begin{align*}
\min_x \max_\alpha x^2 - \alpha(x-b) \\
s.t. \ \alpha \geq 0
\end{align*}
\]
Lagrange multiplier for SVMs

Original formulation

Min \( \frac{w^T w}{2} \)

\((w^T x_i + b) y_i \geq 1\)

Dual formulation

\[
\begin{align*}
\min_{w,b} \quad & \max_{\alpha} \quad \frac{w^T w}{2} - \sum_{i} \alpha_i [(w^T x_i + b) y_i - 1] \\
& \alpha_i \geq 0 \quad \forall i
\end{align*}
\]

Using this new formulation we can derive \( w \) and \( b \) by taking the derivative w.r.t. \( w \) and \( \alpha \) leading to:

\[
\begin{align*}
w &= \sum_{i} \alpha_i x_i y_i \\
b &= y_i - w^T x_i
\end{align*}
\]

for \( i \) s.t. \( \alpha_i > 0 \)

Finally, taking the derivative w.r.t. \( b \) we get:

\[
\sum_{i} \alpha_i y_i = 0
\]
Dual SVM - interpretation

\[ w = \sum_{i} \alpha_i x_i y_i \]

For \( \alpha \)'s that are not 0
Dual SVM for linearly separable case

Substituting \( w \) into our target function and using the additional constraint we get:

**Dual formulation**

\[
\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j
\]

\[
\sum_i \alpha_i y_i = 0
\]

\[
\alpha_i \geq 0 \quad \forall i
\]

\[
\min_{w,b} \max_{\alpha} \frac{w^T w}{2} - \sum_i \alpha_i [(w^T x_i + b) y_i - 1]
\]

\[
\alpha_i \geq 0 \quad \forall i
\]

\[
w = \sum_i \alpha_i x_i y_i
\]

\[
b = y_i - w^T x_i
\]

\[
\text{for } i \ \text{s.t. } \alpha_i > 0
\]

\[
\sum_i \alpha_i y_i = 0
\]
Dual SVM for linearly separable case

Our dual target function:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \quad \forall i$$

To evaluate a new sample $x_k$, we need to compute:

$$w^T x_j + b = \sum_i \alpha_i y_i x_i^T x_k + b$$

Is this too much computational work (for example when using transformation of the data)?
Classifying in 1-d

Can an SVM correctly classify this data?

What about this?
Can an SVM correctly classify this data?

And now?
Non-linear SVMs: 2D

- The original input space \((x)\) can be mapped to some higher-dimensional feature space \((\varphi(x))\) where the training set is separable:

\[
\begin{align*}
  x &= (x_1, x_2) \\
  \varphi(x) &= (x_1^2, x_2^2, \sqrt{2x_1x_2})
\end{align*}
\]
Non-linear SVMs: 2D

- The original input space (x) can be mapped to some higher-dimensional feature space (φ(x)) where the training set is separable:

\[ x = (x_1, x_2) \]

\[ \varphi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \]

If data is mapped into sufficiently high dimension, then samples will in general be linearly separable; N data points are in general separable in a space of N-1 dimensions or more!!!
Transformation of Inputs

• Possible problems
  - High computation burden due to high-dimensionality
  - Many more parameters

• SVM solves these two issues simultaneously
  – “Kernel tricks” for efficient computation
  – Dual formulation only assigns parameters to samples, not features
Quadratic kernels

• While working in higher dimensions is beneficial, it also increases our run time because of the dot product computation.

• However, there is a neat trick we can use.

• Consider all quadratic terms for \( x^1, x^2 \ldots x^m \)

\[
\begin{align*}
\Phi(x) &= \begin{bmatrix}
1 \\
\sqrt{2}x^1 \\
\vdots \\
\sqrt{2}x^m
\end{bmatrix} \\
&= \begin{bmatrix}
(x^1)^2 \\
\vdots \\
(x^m)^2 \\
\sqrt{2}x^1x^2 \\
\vdots \\
\sqrt{2}x^{m-1}x^m
\end{bmatrix}
\end{align*}
\]

\[
\max_{\alpha} \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(x_i) \Phi(x_j)
\]

\[
\sum_i \alpha_i y_i = 0
\]

\[
\alpha_i \geq 0 \quad \forall i
\]

The \( \sqrt{2} \) term will become clear in the next slide.

\( m \) is the number of features in each vector.
Dot product for quadratic kernels

How many operations do we need for the dot product?

$$\Phi(x)\Phi(z) = \begin{pmatrix} 1 & 1 \\ \sqrt{2}x^1 & \sqrt{2}z^1 \\ \vdots & \vdots \\ \sqrt{2}x^1 & \sqrt{2}z^2 \\ (x^1)^2 & (z^1)^2 \\ \vdots & \vdots \\ (x^m)^2 & (x^m)^2 \\ \sqrt{2}x^1x^2 & \sqrt{2}z^1z^2 \\ \vdots & \vdots \\ \sqrt{2}x^{m-1}x^m & \sqrt{2}z^{m-1}z^m \end{pmatrix} = \sum_i 2x^i z^i + \sum_i (x^i)^2 (z^i)^2 + \sum_i \sum_{j=i+1} 2x^i x^j z^i z^j + 1$$

$$= \sum_{i} m_i m + m(m-1)/2 \approx m^2$$
The kernel trick

How many operations do we need for the dot product?

\[
\sum_{i} 2x^i z^i + \sum_{i} (x^i)^2 (z^i)^2 + \sum_{i} \sum_{j=i+1} 2x^i x^j z^i z^j + 1
\]

\[= m \quad m \quad m(m-1)/2 \quad \approx m^2\]

However, we can obtain dramatic savings by noting that

\[(x.z + 1)^2 = (x.z)^2 + 2(x.z) + 1\]

\[= (\sum x^i z^i)^2 + \sum 2x^i z^i + 1\]

\[= \sum 2x^i z^i + \sum (x^i)^2 (z^i)^2 + \sum \sum 2x^i x^j z^i z^j + 1\]

We only need \(m\) operations!

Note that to evaluate a new sample we are also using dot products so we save there as well.
Where we are

Our dual target function:

\[
\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j
\]

\[
\sum_i \alpha_i y_i = 0
\]

\[
\alpha_i \geq 0 \quad \forall i
\]

\[mn^2\] operations at each iteration

To evaluate a new sample \(x_j\) we need to compute:

\[
w^T x_j + b = \sum_i \alpha_i y_i x_i^T x + b
\]

\[mr\] operations where \(r\) are the number of support vectors \((\alpha_i > 0)\)
Other kernels

• The kernel trick works for higher order polynomials as well.

• For example, a polynomial of degree 4 can be computed using 
  \((x.z+1)^4\) and, for a polynomial of degree \(d\) 
  \((x.z+1)^d\)

• Beyond polynomials there are other very high dimensional basis 
  functions that can be made practical by finding the right Kernel 
  Function

- Radial-Basis-style Kernel Function: 
  \[ K(x,z) = \exp\left(-\frac{(x - z)^2}{2\sigma^2}\right) \]

- Neural-net-style Kernel Function: 
  \[ K(x,z) = \tanh(\kappa x.z - \delta) \]
Dual formulation for non linearly separable case

Dual target function:

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j$$

$$\sum_{i} \alpha_i y_i = 0$$

$$C > \alpha_i \geq 0 \quad \forall i$$

The only difference is that the \( \alpha_i \)'s are now bounded

To evaluate a new sample \( x_j \) we need to compute:

$$w^T x_j + b = \sum_{i} \alpha_i y_i x_i x_j + b$$
Why do SVMs work?

• If we are using huge features spaces (with kernels) how come we are not overfitting the data?
  - Number of parameters remains the same (and most are set to 0)
  - While we have a lot of input values, at the end we only care about the support vectors and these are usually a small group of samples
  - The minimization (or the maximizing of the margin) function acts as a sort of regularization term leading to reduced overfitting
Software

• A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
• Some implementation (such as LIBSVM) can handle multi-class classification
• SVMLight is among one of the earliest implementation of SVM
• Several Matlab toolboxes for SVM are also available
Multi-class classification with SVMs

What if we have data from more than two classes?

- Most common solution: One vs. all
  - create a classifier for each class against all other data
  - for a new point use all classifiers and compare the margin for all selected classes

Note that this is not necessarily valid since this is not what we trained the SVM for, but often works well in practice
Applications of SVMs

• Bioinformatics
• Machine Vision
• Text Categorization
• Ranking (e.g., Google searches)
• Handwritten Character Recognition
• Time series analysis

→ Lots of very successful applications!!!
Handwritten digit recognition

3-nearest-neighbor = 2.4% error
400–300–10 unit MLP = 1.6% error
LeNet: 768–192–30–10 unit MLP = 0.9% error

Current best (kernel machines, vision algorithms) ≈ 0.6% error
Important points

• Difference between regression classifiers and SVMs’
• Maximum margin principle
• Target function for SVMs
• Linearly separable and non separable cases
• Dual formulation of SVMs
• Kernel trick and computational complexity