Support Vector Machines & Kernels

Doing really well with linear decision surfaces

Adapted from slides by Tim Oates
Outline

- Prediction
  - Why might predictions be wrong?
- Support vector machines
  - Doing really well with linear models
- Kernels
  - Making the non-linear linear
Why Might Predictions be Wrong?

• True non-determinism
  – Flip a biased coin
  – \( p(\text{heads}) = \theta \)
  – Estimate \( \theta \)
  – If \( \theta > 0.5 \) predict ‘heads’, else ‘tails’

Lots of ML research on problems like this:
  – Learn a model
  – Do the best you can in expectation
Why Might Predictions be Wrong?

• Partial observability
  – Something needed to predict $y$ is missing from observation $x$
  – $N$-bit parity problem
    • $x$ contains $N-1$ bits (hard PO)
    • $x$ contains $N$ bits but learner ignores some of them (soft PO)

• Noise in the observation $x$
  – Measurement error
  – Instrument limitations
Why Might Predictions be Wrong?

- True non-determinism
- Partial observability
  - hard, soft
- Representational bias
- Algorithmic bias
- Bounded resources
Representational Bias

- Having the right features ($x$) is crucial
Support Vector Machines

Doing *Really* Well with Linear Decision Surfaces
Strengths of SVMs

• Good generalization
  – in theory
  – in practice
• Works well with few training instances
• Find globally best model
• Efficient algorithms
• Amenable to the kernel trick
Minor Notation Change

To better match notation used in SVMs
...and to make matrix formulas simpler

We will drop using superscripts for the $i^{\text{th}}$ instance

$i^{\text{th}}$ instance

$x^{(i)} \rightarrow x_i$

$i^{\text{th}}$ instance label

$y^{(i)} \rightarrow y_i$

$j^{\text{th}}$ feature of $i^{\text{th}}$ instance

$x^{(i)}_j \rightarrow x_{ij}$

Bold denotes vector

Non-bold denotes scalar
Linear Separators

• Training instances
  \(\mathbf{x} \in \mathbb{R}^{d+1}, x_0 = 1\)
  \(y \in \{-1, 1\}\)

• Model parameters
  \(\mathbf{\theta} \in \mathbb{R}^{d+1}\)

• Hyperplane
  \(\mathbf{\theta}^\top \mathbf{x} = \langle \mathbf{\theta}, \mathbf{x} \rangle = 0\)

• Decision function
  \(h(\mathbf{x}) = \text{sign}(\mathbf{\theta}^\top \mathbf{x}) = \text{sign}(\langle \mathbf{\theta}, \mathbf{x} \rangle)\)

Recall:
Inner (dot) product:
\[
\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^\top \mathbf{v}
= \sum_i u_i v_i
\]
Intuitions
Intuitions
Intuitions
A “Good” Separator
Noise in the Observations
Ruling Out Some Separators
Lots of Noise
Only One Separator Remains
Maximizing the Margin
“Fat” Separators
Why Maximize Margin

Increasing margin reduces *capacity*

• i.e., fewer possible models

Lesson from Learning Theory:

• If the following holds:
  
  – $H$ is sufficiently constrained in size
  – and/or the size of the training data set $n$ is large,
  
  then low training error is likely to be evidence of low generalization error
Alternative View of Logistic Regression

\[ h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \]

If \( y = 1 \), we want \( h_\theta(x) \approx 1 \), \( \theta^T x \gg 0 \)

If \( y = 0 \), we want \( h_\theta(x) \approx 0 \), \( \theta^T x \ll 0 \)

\[
J(\theta) = - \sum_{i=1}^{n} \left[ y_i \log h_\theta(x_i) + (1 - y_i) \log (1 - h_\theta(x_i)) \right]
\]

\[
\min_\theta J(\theta) = \text{cost}_1(\theta^T x_i) \quad \text{cost}_0(\theta^T x_i)
\]

Based on slide by Andrew Ng
Alternate View of Logistic Regression

Cost of example: 

\[-y_i \log h_\theta(x_i) - (1 - y_i) \log (1 - h_\theta(x_i))\]

\[h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}\]

\[z = \theta^T x\]

If \(y = 1\) (want \(\theta^T x \gg 0\)): 

\[-\log \frac{1}{1 + e^{-z}}\]

If \(y = 0\) (want \(\theta^T x \ll 0\)): 

\[-\log(1 - \frac{1}{1 + e^{-z}})\]
Logistic Regression to SVMs

Logistic Regression:

\[
\min_{\theta} \ - \sum_{i=1}^{n} \left[ y_i \log h_\theta(x_i) + (1 - y_i) \log (1 - h_\theta(x_i)) \right] + \frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2
\]

Support Vector Machines:

\[
\min_{\theta} \ C \sum_{i=1}^{n} \left[ y_i \text{cost}_1(\theta^T x_i) + (1 - y_i) \text{cost}_0(\theta^T x_i) \right] + \frac{1}{2} \sum_{j=1}^{d} \theta_j^2
\]

You can think of \( C \) as similar to \( \frac{1}{\lambda} \).
Support Vector Machine

\[
\min \theta \sum_{i=1}^{n} [y_i \text{cost}_1(\theta^T x_i) + (1 - y_i) \text{cost}_0(\theta^T x_i)] + \frac{1}{2} \sum_{j=1}^{d} \theta_j^2
\]

If \( y = 1 \) (want \( \theta^T x \geq 1 \)):  

If \( y = 0 \) (want \( \theta^T x \leq -1 \)):

\[
\ell_{\text{hinge}}(h(x)) = \max(0, 1 - y \cdot h(x))
\]

Based on slide by Andrew Ng
Support Vector Machine

\[
\min_{\theta} \ C \sum_{i=1}^{n} [y_i \text{cost}_1(\theta^T x_i) + (1 - y_i) \text{cost}_0(\theta^T x_i)] + \frac{1}{2} \sum_{j=1}^{d} \theta_j^2
\]

\[y = 1 / 0\]
\[y = +1 / -1\]

with \( C = 1 \)

\[
\min_{\theta} \ \frac{1}{2} \sum_{j=1}^{d} \theta_j^2
\]

s.t. \( \theta^T x_i \geq 1 \) if \( y_i = 1 \)
\( \theta^T x_i \leq -1 \) if \( y_i = -1 \)
Maximum Margin Hyperplane

\[ \text{margin} = \frac{2}{\|\theta\|_2} \]

\[ \theta^T x = 1 \]

\[ \theta^T x = -1 \]
Support Vectors

\[ \sum \alpha_i y_i \mathbf{x}_i \]

\[ \theta^T \mathbf{x} = 1 \]

\[ \theta^T \mathbf{x} = -1 \]
Large Margin Classifier in Presence of Outliers

Based on slide by Andrew Ng
Vector Inner Product

\[
\begin{align*}
    &u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\
    &\|u\|_2 = \text{length}(u) \in \mathbb{R} \\
    &\quad = \sqrt{u_1^2 + u_2^2} \\
    &u^\top v = v^\top u \\
    &\quad = u_1 v_1 + u_2 v_2 \\
    &\quad = \|u\|_2 \|v\|_2 \cos \theta \\
    &\quad = p \|u\|_2 \quad \text{where } p = \|v\|_2 \cos \theta
\end{align*}
\]
Understanding the Hyperplane

\[
\min_{\theta} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2
\]

s.t. \( \theta^T x_i \geq 1 \) if \( y_i = 1 \)
\( \theta^T x_i \leq -1 \) if \( y_i = -1 \)

Assume \( \theta_0 = 0 \) so that the hyperplane is centered at the origin, and that \( d = 2 \)

\[
\theta^T x = \|\theta\|_2 \|x\|_2 \cos \theta
\]

\[
= p \|\theta\|_2
\]

Based on example by Andrew Ng
Maximizing the Margin

\[
\min_{\theta} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2
\]

s.t. \( \theta^T x_i \geq 1 \) if \( y_i = 1 \)
\( \theta^T x_i \leq -1 \) if \( y_i = -1 \)

Assume \( \theta_0 = 0 \) so that the hyperplane is centered at the origin, and that \( d = 2 \)

Let \( p_i \) be the projection of \( x_i \) onto the vector \( \theta \)

Based on example by Andrew Ng

Since \( p \) is small, therefore \( \|\theta\|_2 \) must be large to have \( p \|\theta\|_2 \geq 1 \) (or \( \leq -1 \))

Since \( p \) is larger, \( \|\theta\|_2 \) can be smaller in order to have \( p \|\theta\|_2 \geq 1 \) (or \( \leq -1 \))
Size of the Margin

For the support vectors, we have $p \|\theta\|_2 = \pm 1$

- $p$ is the length of the projection of the SVs onto $\theta$

Therefore,

$$p = \frac{1}{\|\theta\|_2}$$

$$\text{margin} = 2p = \frac{2}{\|\theta\|_2}$$
The SVM Dual Problem

The primal SVM problem was given as

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2$$

s.t. \( y_i(\theta^T x_i) \geq 1 \ \forall i \)

Can solve it more efficiently by taking the Lagrangian dual

- Duality is a common idea in optimization
- It transforms a difficult optimization problem into a simpler one
- Key idea: introduce slack variables \( \alpha_i \) for each constraint
  - \( \alpha_i \) indicates how important a particular constraint is to the solution
The SVM Dual Problem

- The Lagrangian is given by

\[ L(\theta, \alpha) = \frac{1}{2} \sum_{j=1}^{d} \theta_j^2 - \sum_{i=1}^{n} \alpha_i (y_i \theta^\top x - 1) \]

s.t. \( \alpha_i \geq 0 \ \forall i \)

- We must minimize over \( \theta \) and maximize over \( \alpha \)
- At optimal solution, partials w.r.t \( \theta \)'s are 0

Solve by a bunch of algebra and calculus ...
and we obtain ...
SVM Dual Representation

Maximize \[ J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \]

s.t. \[ \alpha_i \geq 0 \quad \forall i \]

\[ \sum_{i} \alpha_i y_i = 0 \]

The decision function is given by

\[ h(x) = \text{sign} \left( \sum_{i \in SV} \alpha_i y_i \langle x, x_i \rangle + b \right) \]

where \[ b = \frac{1}{|SV|} \sum_{i \in SV} \left( y_i - \sum_{j \in SV} \alpha_j y_j \langle x_i, x_j \rangle \right) \]
Understanding the Dual

Maximize

\[ J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \]

s.t.

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

\[ \sum_{i} \alpha_i y_i = 0 \]

Balances between the weight of constraints for different classes

Constraint weights (\(\alpha_i\)’s) cannot be negative
Intuitively, we should be more careful around points near the margin.
Understanding the Dual

Maximize\[ J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \]

s.t. \( \alpha_i \geq 0 \ \forall i \)

\[ \sum_{i} \alpha_i y_i = 0 \]

In the solution, either:

- \( \alpha_i > 0 \) and the constraint is tight \( (y_i(\theta^T x_i) = 1) \)
  - point is a support vector
- \( \alpha_i = 0 \)
  - point is not a support vector
Empowering the Solution

• Given the optimal solution $\alpha^*$, optimal weights are

$$\theta^* = \sum_{i \in SVs} \alpha_i^* y_i x_i$$

– In this formulation, have not added $x_0 = 1$

• Therefore, we can solve one of the SV constraints

$$y_i (\theta^* \cdot x_i + \theta_0) = 1$$

to obtain $\theta_0$

– Or, more commonly, take the average solution over all support vectors
What if Data Are Not Linearly Separable?

- Cannot find $\theta$ that satisfies $y_i(\theta^T x_i) \geq 1 \; \forall i$

- Introduce slack variables $\xi_i$

$$y_i(\theta^T x_i) \geq 1 - \xi_i \; \forall i$$

- New problem:

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2 + C \sum_i \xi_i$$

s.t. $y_i(\theta^T x_i) \geq 1 - \xi_i \; \forall i$
Strengths of SVMs

- Good generalization in theory
- Good generalization in practice
- Work well with few training instances
- Find globally best model
- Efficient algorithms
- Amenable to the kernel trick ...
What if Surface is Non-Linear?
Kernel Methods

Making the Non-Linear Linear
When Linear Separators Fail

\[ x! \]

\[ x^2 \]
Mapping into a New Feature Space

\[ \Phi : \mathcal{X} \mapsto \hat{\mathcal{X}} = \Phi(x) \]

• For example, with \( x_i \in \mathbb{R}^2 \)
  \[ \Phi([x_{i1}, x_{i2}]) = [x_{i1}, x_{i2}, x_{i1}x_{i2}, x_{i1}^2, x_{i2}^2] \]

• Rather than run SVM on \( x_i \), run it on \( \Phi(x_i) \)
  – Find non-linear separator in input space

• What if \( \Phi(x_i) \) is really big?
• Use kernels to compute it implicitly!

Image from http://web.engr.oregonstate.edu/~afern/classes/cs534/
Kernels

• Find kernel $K$ such that
  \[ K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \]

• Computing $K(x_i, x_j)$ should be efficient, much more so than computing $\Phi(x_i)$ and $\Phi(x_j)$

• Use $K(x_i, x_j)$ in SVM algorithm rather than $\langle x_i, x_j \rangle$

• Remarkably, this is possible!
The Polynomial Kernel

Let \( \mathbf{x}_i = [x_{i1}, x_{i2}] \) and \( \mathbf{x}_j = [x_{j1}, x_{j2}] \)

Consider the following function:

\[
K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^2
= (x_{i1}x_{j1} + x_{i2}x_{j2})^2
= (x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2})
= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle
\]

where

\[
\Phi(\mathbf{x}_i) = [x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2}]
\]

\[
\Phi(\mathbf{x}_j) = [x_{j1}^2, x_{j2}^2, \sqrt{2}x_{j1}x_{j2}]
\]
The Polynomial Kernel

- Given by $K(x_i, x_j) = \langle x_i, x_j \rangle^d$
  - $\Phi(x)$ contains all monomials of degree $d$

- Useful in visual pattern recognition
  - Example:
    - 16x16 pixel image
    - $10^{10}$ monomials of degree 5
    - Never explicitly compute $\Phi(x)$!

- Variation: $K(x_i, x_j) = (\langle x_i, x_j \rangle + 1)^d$
  - Adds all lower-order monomials (degrees $1,\ldots,d$)!
The Kernel Trick

“Given an algorithm which is formulated in terms of a positive definite kernel $K_1$, one can construct an alternative algorithm by replacing $K_1$ with another positive definite kernel $K_2$”

- SVMs can use the kernel trick
Incorporating Kernels into SVM

\[ J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \]

s.t. \( a_i \geq 0 \ \forall i \)

\[ \sum_{i} \alpha_i y_i = 0 \]
The Gaussian Kernel

• Also called Radial Basis Function (RBF) kernel

\[ K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \]

– Has value 1 when \(x_i = x_j\)
– Value falls off to 0 with increasing distance
– Note: Need to do feature scaling before using Gaussian Kernel

\(\sigma^2 = 0.5\)

lower bias,
higher variance

\(\sigma^2 = 1\)

\(\sigma^2 = 3\)

higher bias,
lower variance
Gaussian Kernel Example

\[ K(x_i, x_j) = \exp \left( -\frac{\|x_i - x_j\|^2}{2\sigma^2} \right) \]

Imagine we’ve learned that:

\[ \theta = [-0.5, 1, 1, 0] \]

Predict +1 if \( \theta_0 + \theta_1 K(x, \ell_1) + \theta_2 K(x, \ell_2) + \theta_3 K(x, \ell_3) \geq 0 \)
Gaussian Kernel Example

\[ K(x_i, x_j) = \exp \left( -\frac{||x_i - x_j||^2}{2\sigma^2} \right) \]

Imagine we’ve learned that:

\[ \theta = [-0.5, 1, 1, 0] \]

Predict +1 if \( \theta_0 + \theta_1 K(x, \ell_1) + \theta_2 K(x, \ell_2) + \theta_3 K(x, \ell_3) \geq 0 \)

- For \( x_1 \), we have \( K(x_1, \ell_1) \approx 1 \), other similarities \( \approx 0 \)

\[
\theta_0 + \theta_1(1) + \theta_2(0) + \theta_3(0) \\
= -0.5 + 1(1) + 1(0) + 0(0) \\
= 0.5 \geq 0 \text{, so predict +1}
\]
Gaussian Kernel Example

\[ K(x_i, x_j) = \exp \left( - \frac{\|x_i - x_j\|^2}{2\sigma^2} \right) \]

Imagine we’ve learned that:
\[ \theta = [-0.5, 1, 1, 0] \]

Predict +1 if \[ \theta_0 + \theta_1 K(x, \ell_1) + \theta_2 K(x, \ell_2) + \theta_3 K(x, \ell_3) \geq 0 \]

- For \( x_2 \), we have \( K(x_2, \ell_3) \approx 1 \), other similarities \( \approx 0 \)
\[ \theta_0 + \theta_1 (0) + \theta_2 (0) + \theta_3 (1) \]
\[ = -0.5 + 1(0) + 1(0) + 0(1) \]
\[ = -0.5 < 0 \], so predict -1

Based on example by Andrew Ng
Gaussian Kernel Example

\[ K(x_i, x_j) = \exp \left( -\frac{\|x_i - x_j\|^2}{2\sigma^2} \right) \]

Imagine we’ve learned that:
\[ \theta = [-0.5, 1, 1, 0] \]

Predict +1 if \[ \theta_0 + \theta_1 K(x, \ell_1) + \theta_2 K(x, \ell_2) + \theta_3 K(x, \ell_3) \geq 0 \]

Rough sketch of decision surface

Based on example by Andrew Ng
Other Kernels

• Sigmoid Kernel

\[ K(x_i, x_j) = \tanh(\alpha x_i^T x_j + c) \]

  – Neural networks use sigmoid as activation function
  – SVM with a sigmoid kernel is equivalent to 2-layer perceptron

• Cosine Similarity Kernel

\[ K(x_i, x_j) = \frac{x_i^T x_j}{\|x_i\| \|x_j\|} \]

  – Popular choice for measuring similarity of text documents
  – L_2 norm projects vectors onto the unit sphere; their dot product is the cosine of the angle between the vectors
Other Kernels

• Chi-squared Kernel

\[ K(x_i, x_j) = \exp \left( -\gamma \sum_k \frac{(x_{ik} - x_{jk})^2}{x_{ik} + x_{jk}} \right) \]

– Widely used in computer vision applications
– Chi-squared measures distance between probability distributions
– Data is assumed to be non-negative, often with \( L_1 \) norm of 1

• String kernels
• Tree kernels
• Graph kernels
An Aside: The Math Behind Kernels

What does it mean to be a kernel?
• $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$ for some $\Phi$

What does it take to be a kernel?
• The Gram matrix $G_{ij} = K(x_i, x_j)$
  – Symmetric matrix
  – Positive semi-definite matrix:
    \[ z^T G z \geq 0 \text{ for every non-zero vector } z \in \mathbb{R}^n \]

Establishing “kernel-hood” from first principles is non-trivial
A Few Good Kernels...

- Linear Kernel
  \[ K(x_i, x_j) = \langle x_i, x_j \rangle \]

- Polynomial kernel
  \[ K(x_i, x_j) = (\langle x_i, x_j \rangle + c)^d \]
  \(- c \geq 0 \) trades off influence of lower order terms

- Gaussian kernel
  \[ K(x_i, x_j) = \exp \left( - \frac{\|x_i - x_j\|^2}{2\sigma^2} \right) \]

- Sigmoid kernel
  \[ K(x_i, x_j) = \tanh \left( \alpha x_i^T x_j + c \right) \]

Many more...

- Cosine similarity kernel
- Chi-squared kernel
- String/tree/graph/wavelet/etc kernels
Application: Automatic Photo Retouching

(Leyvand et al., 2008)
Practical Advice for Applying SVMs

• Use SVM software package to solve for parameters
  – e.g., SVMlight, libsvm, cvx (fast!), etc.

• Need to specify:
  – Choice of parameter $C$
  – Choice of kernel function
    • Associated kernel parameters
      
      e.g., $K(x_i, x_j) = (\langle x_i, x_j \rangle + c)^d$

      $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$
Multi-Class Classification with SVMs

- Many SVM packages already have multi-class classification built in
- Otherwise, use one-vs-rest
  - Train $K$ SVMs, each picks out one class from rest, yielding $\theta^{(1)}, \ldots, \theta^{(K)}$
  - Predict class $i$ with largest $\left(\theta^{(i)}\right)^T x$

$y \in \{1, \ldots, K\}$

Based on slide by Andrew Ng
SVMs vs Logistic Regression
(Advice from Andrew Ng)

\[ n = \# \text{training examples} \quad d = \# \text{features} \]

If \( d \) is large (relative to \( n \)) (e.g., \( d > n \) with \( d = 10,000, n = 10-1,000 \))
• Use logistic regression or SVM with a linear kernel

If \( d \) is small (up to 1,000), \( n \) is intermediate (up to 10,000)
• Use SVM with Gaussian kernel

If \( d \) is small (up to 1,000), \( n \) is large (50,000+)
• Create/add more features, then use logistic regression or SVM without a kernel

Neural networks likely to work well for most of these settings, but may be slower to train

Based on slide by Andrew Ng
Other SVM Variations

• nu SVM
  – nu parameter controls:
    • Fraction of support vectors (lower bound) and misclassification rate (upper bound)
    • E.g., $\nu = 0.05$ guarantees that $\geq 5\%$ of training points are SVs and training error rate is $\leq 5\%$
      – Harder to optimize than C-SVM and not as scalable

• SVMs for regression

• One-class SVMs

• SVMs for clustering

...
Conclusion

• SVMs find optimal linear separator
• The kernel trick makes SVMs learn non-linear decision surfaces

• Strength of SVMs:
  – Good theoretical and empirical performance
  – Supports many types of kernels

• Disadvantages of SVMs:
  – “Slow” to train/predict for huge data sets (but relatively fast!)
  – Need to choose the kernel (and tune its parameters)