CS184A/284A
AI in Biology and Medicine
SVM
Machine Learning

- Support Vector Machines
- Lagrangian and Dual
- The Kernel Trick
Linear classifiers

- Which decision boundary is “better”?  
  - Both have zero training error (perfect training accuracy)  
  - But, one of them seems intuitively better…

- How can we quantify “better”, and learn the “best” parameter settings?
One possible answer...

- Maybe we want to maximize our “margin”
- To optimize, relate to model parameters
- Remove “scale invariance”
  - Define class +1 in some region, class −1 in another
  - Make those regions as far apart as possible

We could define such a function:

\[ f(x) = w \cdot x + b \]

\[ f(x) > +1 \text{ in region } +1 \]
\[ f(x) < -1 \text{ in region } -1 \]

Passes through zero in center…

“Support vectors” – data points on margin

Notation change!
\[ \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots \]
\[ b + w_1 x_1 + w_2 x_2 + \ldots \]
Computing the margin width

- Vector $\mathbf{w} = [w_1, w_2, \ldots]$ is perpendicular to the boundaries (why?)

- $\mathbf{w} \cdot \mathbf{x} + b = 0 \quad \& \quad \mathbf{w} \cdot \mathbf{x'} + b = 0 \quad \Rightarrow \quad \mathbf{w} \cdot (\mathbf{x'} - \mathbf{x}) = 0$ : orthogonal
Computing the margin width

- Vector $\mathbf{w} = [w_1, w_2, \ldots]$ is perpendicular to the boundaries.
- Choose $\mathbf{x}^-$ s.t. $f(\mathbf{x}^-) = -1$; let $\mathbf{x}^+$ be the closest point with $f(\mathbf{x}^+) = +1$
  - $\mathbf{x}^+ = \mathbf{x}^- + r \cdot \mathbf{w}$ (why?)
- Closest two points on the margin also satisfy
  
  $$w \cdot x^- + b = -1$$  
  $$w \cdot x^+ + b = +1$$

![Diagram with regions and margin](image)
Computing the margin width

- Vector $\mathbf{w} = [w_1, w_2, \ldots]$ is perpendicular to the boundaries
- Choose $\mathbf{x}^-$ s.t. $f(\mathbf{x}^-) = -1$; let $\mathbf{x}^+$ be the closest point with $f(\mathbf{x}^+) = +1$
  \[ \mathbf{x}^+ = \mathbf{x}^- + r \times \mathbf{w} \]
- Closest two points on the margin also satisfy
  \[ w \cdot \mathbf{x}^- + b = -1 \]
  \[ w \cdot \mathbf{x}^+ + b = +1 \]

\[
\begin{align*}
  w \cdot (\mathbf{x}^- + r\mathbf{w}) + b &= +1 \\
  \Rightarrow r \|\mathbf{w}\|^2 + w \cdot \mathbf{x}^- + b &= +1 \\
  \Rightarrow r \|\mathbf{w}\|^2 - 1 &= +1 \\
  \Rightarrow r &= \frac{2}{\|\mathbf{w}\|^2} \\
  M &= \|\mathbf{x}^+ - \mathbf{x}^-\| = \|r\mathbf{w}\| \\
  &= \frac{2}{\|\mathbf{w}\|^2} \|\mathbf{w}\| = \frac{2}{\sqrt{w^T w}}
\end{align*}
\]
### Maximum margin classifier

- Constrained optimization
  - Get all data points correct
  - Maximize the margin

This is an example of a quadratic program: quadratic cost function, linear constraints

\[
w^* = \arg \max_w \frac{2}{\sqrt{w^T w}}
\]

such that “all data on the correct side of the margin”

**Primal problem:**

\[
w^* = \arg \min_w \sum_j w_j^2
\]

s.t.

\[
y^{(i)} = +1 \Rightarrow w \cdot x^{(i)} + b \geq +1
\]

\[
y^{(i)} = -1 \Rightarrow w \cdot x^{(i)} + b \leq -1
\]

(m constraints)
Maximum margin classifier

- Constrained optimization
  - Get all data points correct
  - Maximize the margin

This is an example of a quadratic program: quadratic cost function, linear constraints

Primal problem:

$$w^* = \arg\min_w \sum_j w_j^2$$

s.t.

$$y^{(i)} (w \cdot x^{(i)} + b) \geq +1$$

(m constraints)
A 1D Example

- Suppose we have three data points:
  - $x = -3, y = -1$
  - $x = -1, y = -1$
  - $x = 2, y = 1$

- Many separating perceptrons, $T[ax+b]$
  - Anything with $ax+b = 0$ between -1 and 2

- We can write the margin constraints:
  - $a(-3) + b < -1 \Rightarrow b < 3a - 1$
  - $a(-1) + b < -1 \Rightarrow b < a - 1$
  - $a(2) + b > +1 \Rightarrow b > -2a + 1$
A 1D Example

• Suppose we have three data points
  \[x = -3, \ y = -1\]
  \[x = -1, \ y = -1\]
  \[x = 2, \ y = 1\]

• Many separating perceptrons, \(T[ax+b]\)
  – Anything with \(ax+b = 0\) between -1 and 2

• We can write the margin constraints
  \[a (-3) + b < -1 \quad \Rightarrow \quad b < 3a - 1\]
  \[a (-1) + b < -1 \quad \Rightarrow \quad b < a - 1\]
  \[a (2) + b > +1 \quad \Rightarrow \quad b > -2a + 1\]

• Ex: \(a = 1, \ b = 0\)
A 1D Example

- Suppose we have three data points
  \( x = -3, \ y = -1 \)
  \( x = -1, \ y = -1 \)
  \( x = 2, \ y = 1 \)

- Many separating perceptrons, \( T[ax+b] \)
  - Anything with \( ax+b = 0 \) between -1 and 2

- We can write the margin constraints
  \[
  a (-3) + b < -1 \quad \Rightarrow \quad b < 3a - 1 \\
  a (-1) + b < -1 \quad \Rightarrow \quad b < a - 1 \\
  a (2) + b > +1 \quad \Rightarrow \quad b > -2a + 1
  \]

- Ex: \( a = 1, \ b = 0 \)
- Minimize \( \|a\| \) \( \Rightarrow \) \( a = .66, \ b = -.33 \)
  - Two data on the margin; constraints “tight”
Machine Learning

- Support Vector Machines
- Lagrangian and Dual
- The Kernel Trick
Lagrangian optimization

- Want to optimize constrained system:
  \[ w^* = \arg \min_{w,b} \sum_j w_j^2 \quad s.t. \quad 1 - y^{(i)}(w \cdot x^{(i)} + b) \leq 0 \]

- Introduce Lagrange multipliers \( \alpha \) (one per constraint)
  \[ \theta^* = \arg \min_\theta \max_{\alpha \geq 0} f(\theta) + \sum_i \alpha_i g_i(\theta) \]
  - Can optimize \( \theta, \alpha \) jointly over a simpler constraint set (initialization easy)
  - For inner max:
    \[ g_i(\theta) \leq 0 : \alpha_i = 0 \]
    \[ g_i(\theta) > 0 : \alpha_i \to +\infty \]
  - Any optimum of the original problem is a saddle point of the new
  - KKT complementary slackness:
    \[ \alpha_i > 0 \Rightarrow g_i(\theta) = 0 \]
Notes on Lagrangian optimization

• Equivalence if alpha fully optimized
• Simple to initialize to valid point
  – Gi may be unsatisfied => if so, penalty grows, encouraging theta to satisfy
• Visualization; valid region?
Optimization

- Use Lagrange multipliers
  - Enforce inequality constraints

\[ w^* = \underset{w}{\arg \min} \max_{\alpha \geq 0} \frac{1}{2} \sum_j w_j^2 + \sum_i \alpha_i \left( 1 - y^{(i)} \left( w \cdot x^{(i)} + b \right) \right) \]

Stationary conditions wrt \( w \):

\[ w^* = \sum_i \alpha_i y^{(i)} x^{(i)} \]

and since any support vector has \( y = wx + b \),

\[ b = \frac{1}{N_{sv}} \sum_{i \in SV} \left( y^{(i)} - w \cdot x^{(i)} \right) \]

Alphas > 0 only on the margin:
“support vectors”
Dual form

- Use Lagrange multipliers
  - Enforce inequality constraints
  - Use solution $w^*$ to write solely in terms of alphas:

$$\max_{\alpha \geq 0} \sum_i \left[ \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)}) \right]$$

s.t. $\sum_{i} \alpha_i y^{(i)} = 0$ (since derivative wrt $b = 0$)

Another quadratic program:
optimize m vars with 1+m (simple) constraints
cost function has $m^2$ dot products

$$w^* = \sum_{i} \alpha_i y^{(i)} x^{(i)}$$

$$b = \frac{1}{Nsv} \sum_{i \in SV} (y^{(i)} - w \cdot x^{(i)})$$
Maximum margin classifier

• What if the data are not linearly separable?
  – Want a large “margin”:
    \[
    \min_w \sum_j w_j^2
    \]
  – Want low error:
    \[
    \min_w \sum_i J(y^{(i)}, w \cdot x^{(i)} + b)
    \]
  – “Soft margin” : introduce slack variables for violated constraints

Assigns “cost” R proportional to distance from margin
Another quadratic program!

\[
\begin{align*}
    w^* &= \arg \min_{w, \epsilon} \sum_j w_j^2 + R \sum_i \epsilon^{(i)} \\
    \text{s.t.} \quad & y^{(i)} (w^T x^{(i)} + b) \geq +1 - \epsilon^{(i)} \quad \text{(violate margin by \quad 2)} \\
    & \epsilon^{(i)} \geq 0
\end{align*}
\]
Soft margin SVM

- Large margin vs. Slack variables
- \( R \) large = hard margin
- \( R \) smaller
  - A few wrong predictions; boundary farther from rest

\[
\begin{align*}
  w^* &= \arg\min_{w, \epsilon} \sum_j w_j^2 + R \sum_i \epsilon^{(i)} \\
  \text{s.t.} \\
  y^{(i)} \left( w^T x^{(i)} + b \right) &\geq +1 - \epsilon^{(i)} \\
  \epsilon^{(i)} &\geq 0
\end{align*}
\]
Maximum margin classifier

- **Soft margin optimization:**
  - For *any* weights $w$, we can choose $\varepsilon$ to satisfy constraints
  
  - Write $\varepsilon^*$ as a function of $w$ (call this $J$) and optimize directly

- $J = \text{distance from the "correct" place}$

\[
J_i = \max[0, 1 - y^{(i)}(w \cdot x^{(i)} + b)]
\]

\[
w^* = \arg \min_w \frac{1}{R} \sum_j w_j^2 + \sum_i J_i(y^{(i)}, w \cdot x^{(i)} + b)
\]

(L2 regularization on the weights)

\[
w^* = \arg \min_{w, \epsilon} \sum_j w_j^2 + R \sum_i \epsilon^{(i)}
\]

\[
y^{(i)}(w^T x^{(i)} + b) \geq +1 - \epsilon^{(i)}
\]

(hinge loss)

\[w \cdot x + b \rightarrow +1\]
Soft margin dual:

\[
\max_{0 \leq \alpha \leq R} \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)})
\]

\[\text{s.t. } \sum_i \alpha_i y^{(i)} = 0\]

\(K_{ij}\) measures "similarity" of \(x_i\) and \(x_j\) (their dot product)

Support vectors now data on or past margin…

Prediction:

\[\hat{y} = w^* \cdot x + b = \sum_i \alpha_i y^{(i)} x^{(i)} \cdot x + b\]

\[w^* = \sum_i \alpha_i y^{(i)} x^{(i)}\]

\[b = \ldots\]

More complicated; can solve e.g. using any \(\alpha \in (0, R)\)
Support Vectors

The *support vectors* are data points $i$ with non-zero weight $\alpha_i$:
- Points with minimum margin (on optimized boundary)
- Points which violate margin constraint, but are still correctly classified
- Points which are misclassified

For all other training data, features have *no impact* on learned weight vector

Support vectors now data on or past margin…

Prediction:

$$\hat{y} = w^* \cdot x + b = \sum_i \alpha_i y^{(i)} x^{(i)} \cdot x + b$$

$$w^* = \sum_i \alpha_i y^{(i)} x^{(i)}$$

$$b = \ldots$$

More complicated; can solve e.g. using any $\alpha \in (0,R)$
Multi-class SVMs

• Use standard multi-class linear prediction, 0/1 loss:

\[ \hat{y} = f(x; \theta) = \arg \max_{y} \theta \cdot \Phi(x, y) \]

\[ \Phi(x, y) = [ \mathbb{1}[y = 0] \Phi(x) \ , \ \mathbb{1}[y = 1] \Phi(x) \ , \ldots ] \]

• Hinge-like loss / slack variable optimization:

\[ w^* = \arg \min_{w,b,\epsilon} \sum_{j} w_j^2 + R \sum_{i} \epsilon(i) \]

\[ w^T \Phi(x^{(i)}, y^{(i)}) - w^T \Phi(x^{(i)}, y) \geq 1 - \epsilon(i) \quad \forall y \neq y^{(i)} \]

• Can introduce class-specific loss function: \( \Delta(y, \hat{y}) \)

\[ w^T \Phi(x^{(i)}, y^{(i)}) - w^T \Phi(x^{(i)}, y) \geq \Delta(y^{(i)}, y) - \epsilon(i) \quad \forall y \neq y^{(i)} \]

  – Reduces to earlier form for 0/1 loss:
  \[ \Delta(y, \hat{y}) = \mathbb{1}[y \neq \hat{y}] \]

  – Again, can optimize as QP (e.g., SMO) or hinge-like loss (e.g., SGD)
Linear SVMs

- So far, looked at linear SVMs:
  - Expressible as linear weights “w”
  - Linear decision boundary

- Dual optimization for a linear SVM:

\[
\max_{0 \leq \alpha \leq R} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)}) \\
\text{s.t.} \sum_i \alpha_i y^{(i)} = 0
\]

- Depend on pairwise dot products:
  - Kij measures “similarity”, e.g., 0 if orthogonal

\[
K_{ij} = x^{(i)} \cdot x^{(j)}
\]
Adding features

• Linear classifier can’t learn some functions

1D example:

Not linearly separable

Add quadratic features

Linearly separable in new features…
Adding features

- Recall: feature function \( \Phi(x) \)
  - Predict using some transformation of original features
    \[
    \hat{y}(x) = \text{sign}\left[ w \cdot \Phi(x) + b \right]
    \]

- Dual form of SVM optimization is:
  \[
  \max_{0 \leq \alpha \leq R} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(x^{(i)}) \Phi(x^{(j)})^T \quad \text{s.t.} \quad \sum_i \alpha_i y^{(i)} = 0
  \]

- For example, quadratic (polynomial) features:
  \[
  \Phi(x) = \begin{pmatrix}
  1 & \sqrt{2}x_1 & \sqrt{2}x_2 & \cdots & x_1^2 & x_2^2 & \cdots & \sqrt{2}x_1x_2 & \sqrt{2}x_1x_3 & \cdots
  \end{pmatrix}
  \]
  - Ignore root-2 scaling for now…
  - Expands “x” to length \( O(n^2) \)
Implicit features

- Need $\Phi(x^{(i)})\Phi(x^{(j)})^T$

$$\Phi(x) = (1 \ \sqrt{2}x_1 \ \sqrt{2}x_2 \ \cdots \ x_1^2 \ x_2^2 \ \cdots \ \sqrt{2}x_1x_2 \ \sqrt{2}x_1x_3 \ \cdots)$$

$$\Phi(a) = (1 \ \sqrt{2}a_1 \ \sqrt{2}a_2 \ \cdots \ a_1^2 \ a_2^2 \ \cdots \ \sqrt{2}a_1a_2 \ \sqrt{2}a_1a_3 \ \cdots)$$

$$\Phi(b) = (1 \ \sqrt{2}b_1 \ \sqrt{2}b_2 \ \cdots \ b_1^2 \ b_2^2 \ \cdots \ \sqrt{2}b_1b_2 \ \sqrt{2}b_1b_3 \ \cdots)$$

$$\Phi(a)^T \Phi(b) = 1 + \sum_j 2a_jb_j + \sum_j a_j^2b_j^2 + \sum_j \sum_{k>j} 2a_ja_kb_jb_k + \cdots$$

$$= (1 + \sum_j a_jb_j)^2$$

Can evaluate dot product in only $O(n)$ computations!
Mercer Kernels

- If $K(x,x')$ satisfies Mercer’s condition:
  \[ \int_a \int_b K(a, b) g(a) g(b) \, da \, db \geq 0 \]

- Then, $K(a, b) = \Phi(a) \cdot \Phi(b)$ for some $\Phi(x)$

- Notably, Phi may be hard to calculate
  - May even be infinite dimensional!
  - Only matters that $K(x,x')$ is easy to compute:
  - Computation always stays $O(m^2)$

For all datasets $X$:
\[ g^T \cdot K \cdot g \geq 0 \]
Some commonly used kernel functions & their shape:

Polynomial

\[ K(a, b) = (1 + \sum_{j} a_j b_j)^d \]
Common kernel functions

- Some commonly used kernel functions & their shape:
  - Polynomial: \[ K(a, b) = (1 + \sum_j a_j b_j)^d \]
  - Radial Basis Functions:
    \[ K(a, b) = \exp\left(-\frac{(a - b)^2}{2\sigma^2}\right) \]
Common kernel functions

- Some commonly used kernel functions & their shape:

  - Polynomial: \( K(a, b) = (1 + \sum_j a_j b_j)^d \)
  
  - Radial Basis Functions
    \[ K(a, b) = \exp\left(-\frac{(a-b)^2}{2\sigma^2}\right) \]
  
  - Saturating, sigmoid-like:
    \[ K(a, b) = \tanh(ca^Tb + h) \]
Common kernel functions

• Some commonly used kernel functions & their shape:

  • Polynomial

    \[ K(a, b) = (1 + \sum_{j} a_j b_j)^d \]

  • Radial Basis Functions

    \[ K(a, b) = \exp\left(-\frac{(a - b)^2}{2\sigma^2}\right) \]

  • Saturating, sigmoid-like:

    \[ K(a, b) = \tanh(ca^T b + h) \]

  • Many for special data types:
    – String similarity for text, genetics

  • In practice, may not even be Mercer kernels…
Support Vectors for Kernel SVMs

Support vectors (green) for data separable by radial basis function kernels, and non-linear margin boundaries.
How Many Support Vectors?

Only need to evaluate kernel at support vectors, not all training data. But there may still be a lot of support vectors.
Kernel SVMs

• Linear SVMs
  – Can represent classifier using \( (w,b) = n+1 \) parameters
  – Or, represent using support vectors, \( x^{(i)} \)

• Kernelized?
  – \( K(x,x') \) may correspond to high (infinite?) dimensional \( \Phi(x) \)
  – Typically more efficient to remember the SVs
  – “Instance based” – save data, rather than parameters

• Contrast:
  – Linear SVM: identify features with linear relationship to target
  – Kernel SVM: identify similarity measure between data
    (Sometimes one may be easier; sometimes the other!)
Kernel Least-squares Linear Regression

• Recall L2-regularized linear regression:

\[
\theta = y X (X^T X + \alpha I)^{-1}
\]

Rearranging,

\[
\Rightarrow \theta (X^T X + \alpha I) = y X \\
\alpha \theta = (y - \theta X^T) X
\]

Define:

\[
r = \frac{1}{\alpha} (y - \theta X^T)
\]

\[
\alpha r = y - \theta X^T = y - r X X^T
\]

Gram matrix: \( m \times m \),

\[
K_{ij} = \langle x^{(i)}, x^{(j)} \rangle
\]

Rearrange & solve for \( r \):

\[
r = (X X^T + \alpha I)^{-1} y = (K + \alpha I)^{-1} y
\]

Linear prediction:

\[
\tilde{y} = \langle \theta, \tilde{x} \rangle = r X (\tilde{x})^T = \sum_j r_j \langle x^{(j)}, \tilde{x} \rangle = \sum_j r_j K(x^{(j)}, \tilde{x})
\]

Now just replace \( K(x,x') \) with your desired kernel function!
Example: Kernel Linear Regression

- **K**: MxM

\[ r = \left( K + \alpha I \right)^{-1} y \]

\[ \tilde{y} = \sum_j r_j K(x^{(j)}, \tilde{x}) \]

Linear kernel:

\[ K(x, x') = x^T \cdot x' \]

Gaussian (RBF) kernel:

\[ K(x, x') = \exp \left( -\gamma (x - x')^2 \right) \]
Summary

• Support vector machines

• “Large margin” for separable data
  – Primal QP: maximize margin subject to linear constraints
  – Lagrangian optimization simplifies constraints
  – Dual QP: m variables; involves $m^2$ dot product

• “Soft margin” for non-separable data
  – Primal form: regularized hinge loss
  – Dual form: m-dimensional QP

• Kernels
  – Dual form involves only pairwise similarity
  – Mercer kernels: dot products in implicit high-dimensional space