Instructor: Joelle Pineau (jpineau@cs.mcgill.ca)

Class web page: www.cs.mcgill.ca/~jpineau/comp551
Today’s quiz

Apply Support Vector Machines to data generated by the AND boolean function:

\[ Y = X_1 \text{ AND } X_2 \]

1. What is \( M \), the margin size?

2. What are the weights, \( w^* \)?

3. Which datapoints define the margin?

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>+1</td>
</tr>
</tbody>
</table>
SVM formulation

- SVM problem: \( \text{Min} \quad \frac{1}{2} \| \mathbf{w} \|^2 \)
  w.r.t. \( \mathbf{w} \)
  s.t. \( y_i \mathbf{w}^T \mathbf{x}_i \geq 1 \)

- This can be solved with quadratic programming.
Non-linearly separable data

- A linear boundary might be too simple to capture the data.

- Option 1: **Relax the constraints** and allow some points to be misclassified by the margin.

- Option 2: **Allow a nonlinear decision boundary** in the input space by finding a linear decision boundary in an expanded space (similar to adding polynomial terms in linear regression.)
  - Here $x_i$ is replaced by $\phi(x_i)$, where $\phi$ is called a feature mapping.
Soften the primal objective

- We wanted to solve: \( \min_w \frac{1}{2} \|w\|^2 \)
  \[ \begin{align*}
  \text{s.t.} \quad & y_iw^Tx_i \geq 1 \\
  \end{align*} \]

  This can be re-written: \( \min_w \sum_i L_{0-\infty} (w^Tx_i, y_i) + \frac{1}{2} \|w\|^2 \)
  \[ \begin{align*}
  \text{s.t.} \quad & y_iw^Tx_i \geq 1 \\
  \end{align*} \]

  where \( \sum_i L_{0-\infty} (w^Tx_i, y_i) = (\infty \text{ for a misclassification, } 0 \text{ correct classification}) \)

  Soften misclassification cost: \( \min_w \sum_i L_{0-1} (w^Tx_i, y_i) + \frac{1}{2} \|w\|^2 \)
  \[ \begin{align*}
  \text{s.t.} \quad & y_iw^Tx_i \geq 1 \\
  \end{align*} \]

  where \( \sum_i L_{0-1} (w^Tx_i, y_i) = (1 \text{ for a misclassification, } 0 \text{ correct classification}) \)

  But this is a non-convex objective!
Approximation of the $L_{0-1}$ function

$$l_{0-1}(y, \langle w, x \rangle) = \begin{cases} 1 : y \langle w, x \rangle < 0 \\ 0 : y \langle w, x \rangle > 0 \end{cases}$$

$\text{loss}$

$-2.0$  $-1.5$  $-1.0$  $-0.5$  $0.0$  $0.5$  $1.0$

$-yf(x)$

$0$  $1$  $2$  $3$  $4$

$\text{loss}$

Picture is taken from R. Herbrich.
SVM with hinge loss

- Hinge loss: $L_{hin}(w^T x_i, y_i) = \max \{1 - y_i w^T x_i, 0\}$

- Soften misclassification cost: $\min_w C \sum_i L_{hin}(w^T x_i, y_i) + \frac{1}{2} \|w\|^2$
  where $C$ controls trade-off between slack penalty and margin.

- The hinge loss upper-bounds the 0-1 loss.
  $\xi_i \geq 1 - y_i w^T x_i \geq L_{0-1}(w^T x_i, y_i)$
Primal Soft SVM problem

• Define slack variables \( \xi_i = L_{hin}(w^T x_i, y_i) = \max \{1-y_i w^T x_i, 0\} \)

• Solve: \( \hat{w}_{soft} = \arg\min_{w, \xi} C \sum_{i:1:n} \xi_i + \frac{1}{2}\|w\|^2 \)
  \[ \text{s. t. } y_i w^T x_i \geq 1 - \xi_i, \quad i = 1, \ldots, n \]
  \[ \xi_i \geq 0, \quad i = 1, \ldots, n \]
where \( w \in \mathbb{R}^m, \xi \in \mathbb{R}^n \)

• Introduce Lagrange multipliers: \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T, 0 \leq \alpha_i \)
  \( \beta = (\beta_1, \beta_2, \ldots, \alpha_n)^T, 0 \leq \beta_i \)
Soft SVM problem: Adding Lagrange multipliers

- **Primal** objective: \((w, \xi, \alpha, \beta) = \arg \min_{w,\xi} \max_{\alpha,\beta} L(w, \xi, \alpha, \beta)\)

  where \(L(w, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C \sum_{i:1:n} \xi_i - \sum_{i:1:n} \alpha_i \left(y_i w^T x_i - 1 + \xi_i\right) - \sum_{i:1:n} \beta_i \xi_i\)
Soft SVM problem: Adding Lagrange multipliers

- **Primal** objective: \((w, \xi, \alpha, \beta) = \arg \min_{w,\xi} \max_{\alpha,\beta} L(w, \xi, \alpha, \beta)\)

  where \(L(w, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C \sum_{i=1:n} \xi_i - \sum_{i=1:n} \alpha_i (y_i w^T x_i - 1 + \xi_i) - \sum_{i=1:n} \beta_i \xi_i\)

- **Dual** (invert min and max): \((w, \xi, \alpha, \beta) = \arg \max_{\alpha,\beta} \min_{w,\xi} L(w, \xi, \alpha, \beta)\)
Soft SVM problem: Adding Lagrange multipliers

- **Primal** objective: \((w, \xi, \alpha, \beta) = \arg \min_{w,\xi} \max_{\alpha,\beta} L(w, \xi, \alpha, \beta)\)

  where \(L(w, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C \sum_{i=1:n} \xi_i - \sum_{i=1:n} \alpha_i (y_i w^T x_i - 1 + \xi_i) - \sum_{i=1:n} \beta_i \xi_i\)

- **Dual** (invert min and max): \((w, \xi, \alpha, \beta) = \arg \max_{\alpha,\beta} \min_{w,\xi} L(w, \xi, \alpha, \beta)\)

  Solve: \(\frac{\delta L}{\delta w} = w - \sum_i \alpha_i y_i x_i = 0 \quad \Rightarrow \quad w^* = \sum_i \alpha_i y_i x_i\)

  \(\frac{\delta L}{\delta \xi} = C1_n - \alpha - \beta = 0 \quad \Rightarrow \quad \beta = C1_n - \alpha\)

  Lagrange multipliers are positive, so we have: \(0 \leq \beta_i, \ 0 \leq \alpha_i \leq C\)
Soft SVM problem: Adding Lagrange multipliers

- **Primal** objective: \((w, \xi, \alpha, \beta) = \arg \min_{w,\xi} \max_{\alpha,\beta} L(w, \xi, \alpha, \beta)\)

  where \(L(w, \xi, \alpha, \beta) = \frac{1}{2}||w||^2 + C \sum_{i=1:n} \xi_i - \sum_{i=1:n} \alpha_i (y_i w^T x_i - 1 + \xi_i) - \sum_{i=1:n} \beta_i \xi_i\)

- **Dual** (invert min and max): \((w, \xi, \alpha, \beta) = \arg \max_{\alpha,\beta} \min_{w,\xi} L(w, \xi, \alpha, \beta)\)

  Solve: 
  \[
  \frac{\delta L}{\delta w} = w - \sum_i \alpha_i y_i x_i = 0 \quad \Rightarrow \quad w^* = \sum_i \alpha_i y_i x_i \\
  \frac{\delta L}{\delta \xi} = C 1_n - \alpha - \beta = 0 \quad \Rightarrow \quad \beta = C 1_n - \alpha
  \]

  Lagrange multipliers are positive, so we have: \(0 \leq \beta_i, 0 \leq \alpha_i \leq C\)

- Plug into dual: \(\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)\)

  with constraints \(0 \leq \alpha_i \leq C\) and \(\sum_i \alpha_i y_i = 0\).

- This is a quadratic programming problem (similar to Hard SVM).
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \Rightarrow \infty$, then Soft-SVM $\Rightarrow$ Hard-SVM.
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C\rightarrow\infty$, then Soft-SVM=>Hard-SVM.

- Points away from margin have $\alpha_i = 0$. 
- Points on the margin have $\alpha_i > 0$ and $\xi_i=0$. 
- Points within the margin have $0 < \xi_i < 1$. 
- Points on the decision line have $\xi_i = 1$. 
- Misclassified points have $\xi_i > 1$. 
Soft SVM solution

• Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
• When $C \to \infty$, then Soft-SVM $\to$ Hard-SVM.

• Points away from margin have $\alpha_i = 0$.
• Points on the margin have $\alpha_i > 0$ and $\xi_i = 0$.
• Points within the margin have $0 < \xi_i < 1$.
• Points on the decision line have $\xi_i = 1$.
• Misclassified points have $\xi_i > 1$. 
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \Rightarrow \infty$, then Soft-SVM $\Rightarrow$ Hard-SVM.

- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i = 0$.
- Points within the margin have $0 < \xi_i < 1$.
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$. 

\[ \alpha_j > 0, \quad \xi_j = 0 \]
\[ \alpha_j = 1 \]
\[ 0 < \xi_j < 1 \]
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \rightarrow \infty$, then Soft-SVM=>Hard-SVM.
- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i = 0$.
- Points within the margin have $0 < \xi_i < 1$.
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.

\[ 0 < \xi_i < 1 \]
\[ \alpha_i > 0, \xi_i = 0 \]
\[ \alpha_i = 1 \]
\[ \xi_i = 1 \]
Soft SVM solution

• Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
• When $C \Rightarrow \infty$, then Soft-SVM => Hard-SVM.
• Points away from margin have $\alpha_i = 0$.
• Points on the margin have $\alpha_i > 0$ and $\xi_i=0$.
• Points within the margin have $0 < \xi_i < 1$.
• Points on the decision line have $\xi_i = 1$.
• Misclassified points have $\xi_i > 1$. 

![Diagram showing classification based on $\alpha_i$, $\xi_i$, and decision line](attachment:image.png)
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \to \infty$, then Soft-SVM $\Rightarrow$ Hard-SVM.

- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i = 0$.
- Points within the margin have $0 < \xi_i < 1$.
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.

- To predict on test data:
  \[ h_w(x) = \text{sign}( \sum_{i=1:n} \alpha_i y_i (x_i \cdot x) ) \]
- Only need to store the support vectors (i.e. points on the margin) to predict.
Multiple classes

• **One-vs-All:** Learn K separate binary classifiers.
  
  – Can lead to inconsistent results.
  
  – Training sets are imbalanced, e.g. assuming n examples per class, each binary classifier is trained with positive class having 1*n of the data, and negative class having (K-1)*n of the data.
Multiple classes

• One-vs-All: Learn K separate binary classifiers.
  – Can lead to inconsistent results.
  – Training sets are imbalanced, e.g. assuming n examples per class, each binary classifier is trained with positive class having 1*n of the data, and negative class having (K-1)*n of the data.

• Multi-class SVM: Define the margin to be the gap between the correct class and the nearest other class.
SVMs for regression

- Minimize a regularized error function:
  \[ \hat{w} = \arg\min_w \ C \sum_{i=1:n} (y_i - w^T x_i)^2 + \frac{1}{2}||w||^2 \]

- Introduce slack variables to optimize “tube” around the regression function.
SVMs for regression

- Minimize a regularized error function:
  \[
  \hat{w} = \arg\min_w C \sum_{i:1:n} (y_i - w^T x_i)^2 + \frac{1}{2}||w||^2
  \]

- Introduce slack variables to optimize “tube” around the regression function.

- Typically, relax to $\epsilon$-sensitive error on the linear target to ensure sparse solution (i.e. few support vectors):
  \[
  \hat{w} = \arg\min_w C \sum_{i:1:n} E_\epsilon (y_i - w^T x_i)^2 + \frac{1}{2}||w||^2
  \]
  where \( E_\epsilon = 0 \) if \((y_i - w^T x_i) < \epsilon\),
  \[
  (y_i - w^T x_i) - \epsilon \quad \text{otherwise}
  \]
Non-linearly separable data

• A linear boundary might be too simple to capture the data.

• Option 1: Relax the constraints and allow some points to be misclassified by the margin.

• Option 2: **Allow a nonlinear decision boundary** in the input space by finding a linear decision boundary in an expanded space (*similar to adding polynomial terms in linear regression.*)
  – Here $x_i$ is replaced by $\phi(x_i)$, where $\phi$ is called a feature mapping.
Margin optimization in feature space

- Replacing $x_i$ by $\phi(x_i)$, the optimization problem for $w$ becomes:

  \begin{align*}
  &\text{Min} & \frac{1}{2} \|w\|^2 \\
  \text{w.r.t.} & & w \\
  \text{s.t.} & & y_iw^T\phi(x_i) \geq 1
  \end{align*}
Margin optimization in feature space

• Replacing $x_i$ by $\phi(x_i)$, the optimization problem for $w$ becomes:

  – Primal form: $Min \quad \frac{1}{2} ||w||^2$
    w.r.t. $w$
    s.t. $y_i w^T \phi(x_i) \geq 1$

  – Dual form: $Max \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\phi(x_i) \cdot \phi(x_j))$
    w.r.t. $\alpha_i$
    s.t. $\alpha_i \geq 0$
    $\sum_i \alpha_i y_i = 0$
Feature space solution

- The optimal weights, in the expended feature space, are
  \[ w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \]

- Classification of an input \( x \) is given by:
  \[ h_w(x) = \text{sign}( \sum_{i=1}^{n} \alpha_i y_i (\phi(x_i) \cdot \phi(x)) ) \]
Feature space solution

- The optimal weights, in the expended feature space, are
  \[ \mathbf{w} = \sum_{i=1:n} \alpha_i y_i \phi(\mathbf{x}_i) \]

- Classification of an input \( \mathbf{x} \) is given by:
  \[ h_{\mathbf{w}}(\mathbf{x}) = \text{sign} \left( \sum_{i=1:n} \alpha_i y_i (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})) \right) \]

- Note that to solve the SVM optimization problem in dual form and to make a prediction, we only ever need to compute dot-products of feature vectors.
Kernel functions

• Whenever a learning algorithm (such as SVMs) can be written in terms of dot-products, it can be generalized to kernels.

• A kernel is any function $K: \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$, which corresponds to a dot product for some feature mapping $\phi$:

$$K(x_1, x_2) = \phi(x_1) \cdot \phi(x_2)$$ for some $\phi$
Kernel functions

• Whenever a learning algorithm (such as SVMs) can be written in terms of dot-products, it can be generalized to kernels.

• A kernel is any function $K: \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$, which corresponds to a dot product for some feature mapping $\phi$:

$$K(x_1, x_2) = \phi(x_1) \cdot \phi(x_2)$$

for some $\phi$.

• Conversely, by choosing feature mapping $\phi$, we implicitly choose a kernel function.

• Recall that $\phi(x_1) \cdot \phi(x_2) = \cos \angle(x_1, x_2)$, where $\angle$ denotes the angle between the vectors, so a kernel function can be thought of as a notion of similarity.
Example: Quadratic kernel

- Let $K(x, z) = (x \cdot z)^2$.

- Is this a kernel?

\[
K(x, z) = \left( \sum_{i=1}^{m} x_i z_i \right) \left( \sum_{j=1}^{m} x_j z_j \right) = \sum_{i,j \in \{1..m\}} x_i z_i x_j z_j = \sum_{i,j \in \{1..m\}} (x_i x_j) (z_i z_j)
\]
Example: Quadratic kernel

• Let $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^2$.

• Is this a kernel?

$$K(\mathbf{x}, \mathbf{z}) = (\sum_{i=1:m} x_i z_i) (\sum_{j=1:m} x_j z_j)$$

$$= \sum_{i,j \in \{1..m\}} x_i z_i x_j z_j$$

• We see it is a kernel, with feature mapping:

$$\phi(\mathbf{x}) = \langle x_1^2, x_1 x_2, \ldots, x_1 x_m, x_2 x_1, x_2^2, \ldots, x_m^2 \rangle$$

Feature vector includes all squares of elements and all cross terms.
Example: Quadratic kernel

- Let $K(x, z) = (x \cdot z)^2$.

- Is this a kernel?

$$K(x, z) = (\sum_{i=1:m} x_i z_i) (\sum_{j=1:m} x_j z_j)$$

$$= \sum_{i,j \in \{1..m\}} x_i z_i x_j z_j$$

We see it is a kernel, with feature mapping:

$$\phi(x) = < x_1^2, x_1 x_2, \ldots, x_1 x_m, x_2 x_1, x_2^2, \ldots, x_m^2 >$$

Feature vector includes all squares of elements and all cross terms.

**Important**: Computing $\phi$ takes $O(m^2)$ but computing $K$ only takes $O(m)$. 
Polynomial kernels

- More generally, $K(x, z) = (x \cdot z)^d$ is a kernel, for any positive integer $d$:
  $$K(x, z) = ( \sum_{i=1}^{m} x_i z_i )^d$$
- If we expanded the sum above in the naïve way, we get $n^d$ terms.
- Terms are monomials (products of $x_i$) with total power equal to $d$. 
Polynomial kernels

- More generally, $K(x, z) = (x \cdot z)^d$ is a kernel, for any positive integer $d$:
  $$ K(x, z) = (\sum_{i=1}^{m} x_i z_i)^d $$

- If we expanded the sum above in the naïve way, we get $n^d$ terms.
- Terms are monomials (products of $x_i$) with total power equal to $d$.

- If we use the primal form of the SVM, each term gets a weight.
- **Curse of dimensionality**: it is very expensive both to optimize and to predict with an SVM in primal form.
- However, evaluating the dot-produce of any two feature vectors can be done using $K$ in $O(m)$. 
The “kernel trick”

• If we work with the dual, we do not have to ever compute the feature mapping $\phi$. We just compute the similarity kernel $K$. 
The “kernel trick”

- If we work with the dual, we do not have to ever compute the feature mapping $\phi$. We just compute the similarity kernel $K$.

- We can solve the dual for the $\alpha_i$:

$$\begin{align*}
\text{Max} & \quad \sum_{i=1:n} \alpha_i - \frac{1}{2} \sum_{i,j=1:n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\
w.r.t. & \quad \alpha_i \\
s.t. & \quad \alpha_i \geq 0 \text{ and } \sum_{i=1..n} \alpha_i y_i = 0
\end{align*}$$
The “kernel trick”

- If we work with the dual, we do not have to ever compute the feature mapping $\phi$. We just compute the similarity kernel $K$.

- We can solve the dual for the $\alpha_i$:

$$\text{Max} \quad \sum_{i=1:n} \alpha_i - \frac{1}{2} \sum_{i,j=1:n} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

w.r.t. $\alpha_i$

s.t. $\alpha_i \geq 0$ and $\sum_{i:1..n} \alpha_i y_i = 0$

- The class of a new input $x$ is computed as:

$$h_w(x) = \text{sign}(\sum_{i=1:n} \alpha_i y_i K(x_i, x))$$

where $x_i$ are the support vectors (defining the margin).

- Remember, $K(\cdot, \cdot)$ can be evaluated in $O(m)$ time = big savings!
Some other kernel functions

• $K(x, z) = (1 + x \cdot z)^d$ - feature expansion has all monomial terms of total power.

• Radial basis / Gaussian kernel: $K(x, z) = \exp \left( -\frac{|x-z|^2}{2\sigma^2} \right)$
  
  – This kernel has an infinite-dimensional feature expansion, but dot-products can still be computed in $O(m)$ (where $m=\#\text{features}$)

• Sigmoidal kernel: $K(x, z) = \tanh(c_1 x \cdot z + c_2)$
Example: Gaussian kernel

Note the non-linear decision boundary

$K(x, z) = (1 + x \cdot z)^d$ – feature expansion has all monomial terms of total power $d$.

The kernel has an infinite-dimensional feature expansion, but dot products can still be computed in $O(n)$.

• Radial basis Gaussian kernel:

  $K(x, z) = e^{\frac{-x \cdot z}{2\sigma^2}}$

  The kernel has an infinite-dimensional feature expansion, but dot products can still be computed in $O(n)$.

• Sigmoidal kernel:

  $K(x, z) = \tanh(c_1 x \cdot z + c_2)$

  Note the non-linear decision boundary.
Kernels beyond SVMs

• A lot of research related to defining kernel functions suitable to particular tasks / kinds of inputs (e.g. words, graphs, images).

• Many kernels are available:
  – Information diffusion kernels (Lafferty and Lebanon, 2002)
  – Diffusion kernels on graphs (Kondor and Jebara, 2003)
  – String kernels for text classification (Lodhi et al, 2002)
  – String kernels for protein classification (Leslie et al, 2002)
… and others!
Example: String kernels

• Very important for DNA matching, text classification, …

• Often use a sliding window of length $k$ over the two strings that we want to compare.

  • Within the fixed-size window we can do many things:
    - Count exact matches.
    - Weigh mismatches based on how bad they are.
    - Count certain markers, e.g. AGT.

• The kernel is the sum of these similarities over the two sequences.
Kernelizing other ML algorithms

• Many other machine learning algorithms have a “dual formulation”, in which dot-products of features can be replaced by kernels.

• Examples:
  – Perceptron
  – Logistic regression
  – Linear regression
What you should know

From last class and from today:

- Perceptron algorithm.
- Margin definition for linear SVMs.
- Use of Lagrange multipliers to transform optimization problems.
- Primal and dual optimization problems for SVMs.
- Feature space version of SVMs.
- The kernel trick and examples of common kernels.