Support Vector Machines (SVMs)

Based on slides by Daniel Lowd, Doina Precup and others
Binary Classification Revisited

- Consider a linearly separable binary classification data set \( \{x_i, y_i\}_{i=1}^m \).
- There is an infinite number of hyperplanes that separate the classes:

- Which plane is best?
- Relatedly, for a given plane, for which points should we be most confident in the classification?
The margin, and linear SVMs

- For a given separating hyperplane, the *margin* is two times the (Euclidean) distance from the hyperplane to the nearest training example.

- It is the width of the “strip” around the decision boundary containing no training examples.

- A linear SVM is a perceptron for which we choose $w, w_0$ so that margin is maximized.
Distance to the decision boundary

- Suppose we have a decision boundary that separates the data.

- Let $\gamma_i$ be the distance from instance $x_i$ to the decision boundary.
- How can we write $\gamma_i$ in term of $x_i, y_i, w, w_0$?
Distance to the decision boundary

- The vector $\mathbf{w}$ is normal to the decision boundary. Thus, $\frac{\mathbf{w}}{||\mathbf{w}||}$ is the unit normal.
- The vector from the B to A is $\gamma_i \frac{\mathbf{w}}{||\mathbf{w}||}$.
- B, the point on the decision boundary nearest $\mathbf{x}_i$, is $\mathbf{x}_i - \gamma_i \frac{\mathbf{w}}{||\mathbf{w}||}$.
- As B is on the decision boundary,

$$\mathbf{w} \cdot \left( \mathbf{x}_i - \gamma_i \frac{\mathbf{w}}{||\mathbf{w}||} \right) + w_0 = 0$$

- Solving for $\gamma_i$ yields, for a positive example:

$$\gamma_i = \frac{\mathbf{w}}{||\mathbf{w}||} \cdot \mathbf{x}_i + \frac{w_0}{||\mathbf{w}||}$$
The margin

- The *margin of the hyperplane* is $2M$, where $M = \min_i \gamma_i$
- The most direct statement of the problem of finding a maximum margin separating hyperplane is thus

$$
\max_{w, w_0} \min_i \gamma_i
$$

$$
\equiv \max_{w, w_0} \min_i y_i \left( \frac{w}{\|w\|} \cdot x_i + \frac{w_0}{\|w\|} \right)
$$

- This turns out to be inconvenient for optimization, however. . .
Treating $\gamma_i$ as the constraints

- From the definition of margin, we have:

$$M \leq \gamma_i = y_i \left( \frac{w}{\|w\|} \cdot x_i + \frac{w_0}{\|w\|} \right) \quad \forall i$$

- This suggests:

  maximize $M$

  with respect to $w, w_0$

  subject to $y_i \left( \frac{w}{\|w\|} \cdot x_i + \frac{w_0}{\|w\|} \right) \geq M$ for all $i$

- Problems:
  - $w$ appears nonlinearly in the constraints.
  - This problem is underconstrained. If $(w, w_0, M)$ is an optimal solution, then so is $(\beta w, \beta w_0, M)$ for any $\beta > 0$. 
Adding a constraint

- Let’s try adding the constraint that $\|w\|_M = 1$.
- This allows us to rewrite the objective function and constraints as:
  \[
  \min_{w, w_0} \|w\| \\
  \text{w.r.t. } w, w_0 \\
  \text{s.t. } y_i(w \cdot x_i + w_0) \geq 1
  \]
- This is really nice because the constraints are linear.
- The objective function $\|w\|$ is still a bit awkward.
Final formulation

- Let’s maximize $\|\mathbf{w}\|^2$ instead of $\|\mathbf{w}\|$.  
  (Taking the square is a monotone transformation, as $\|\mathbf{w}\|$ is positive, so this doesn’t change the optimal solution.)

- This gets us to:
  
  $$\begin{align*}
  \text{min} & \quad \|\mathbf{w}\|^2 \\
  \text{w.r.t.} & \quad \mathbf{w}, w_0 \\
  \text{s.t.} & \quad y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1
  \end{align*}$$

- This we can solve! How?
  - It is a quadratic programming (QP) problem—a standard type of optimization problem for which many efficient packages are available.
  - Better yet, it’s a convex (positive semidefinite) QP

https://en.wikipedia.org/wiki/Quadratic_programming
Quadratic programming

The quadratic programming problem with $n$ variables and $m$ constraints can be formulated as follows.\[1\] Given:

- a real-valued, $n$-dimensional vector $c$,
- an $n \times n$-dimensional real symmetric matrix $Q$,
- an $m \times n$-dimensional real matrix $A$, and
- an $m$-dimensional real vector $b$,

the objective of quadratic programming is to find an $n$-dimensional vector $x$, that will

\[
\text{minimize } \frac{1}{2} x^T Q x + c^T x
\]

subject to $Ax \leq b$,

[1]: https://en.wikipedia.org/wiki/Quadratic_programming
Example

\begin{itemize}
\item \( w = \begin{bmatrix} 49.6504 & 46.8962 \end{bmatrix} \) \quad \( w_0 = -48.6936 \)
\item \( w = \begin{bmatrix} 11.7959 & 12.8066 \end{bmatrix} \) \quad \( w_0 = -12.9174 \)
\end{itemize}
Lagrange multipliers for inequality constraints

- Suppose we have the following optimization problem, called *primal*:

\[
\min_w \ f(w)
\]

such that \( g_i(w) \leq 0, \ i = 1 \ldots k \)

- We define the *generalized Lagrangian*:

\[
L(w, \alpha) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w), \tag{1}
\]

where \( \alpha_i, \ i = 1 \ldots k \) are the Lagrange multipliers.
A different formalization

- Consider $P(w) = \max_{\alpha: \alpha_i \geq 0} L(w, \alpha)$
- Observe that the follow is true (see extra notes):

$$P(w) = \begin{cases} f(w) & \text{if all constraints are satisfied} \\ +\infty & \text{otherwise} \end{cases}$$

- Hence, instead of computing $\min_w f(w)$ subject to the original constraints, we can compute:

$$p^* = \min_w P(w) = \min_w \max_{\alpha: \alpha_i \geq 0} L(w, \alpha)$$
Dual optimization problem

- Let $d^* = \max_{\alpha: \alpha_i \geq 0} \min_w L(w, \alpha)$ (max and min are reversed).
- We can show that $d^* \leq p^*$.
  - Let $p^* = L(w^p, \alpha^p)$
  - Let $d^* = L(w^d, \alpha^d)$
  - Then $d^* = L(w^d, \alpha^d) \leq L(w^p, \alpha^d) \leq L(w^p, \alpha^p) = p^*$. 
Dual optimization problem

- If $f$, $g_i$ are convex and the $g_i$ can all be satisfied simultaneously for some $w$, then we have equality: $d^* = p^* = L(w^*, \alpha^*)$

- Moreover $w^*, \alpha^*$ solve the primal and dual if and only if they satisfy the following conditions (called Karush-Kuhn-Tucker):

\[
\frac{\partial}{\partial w_i} L(w^*, \alpha^*) = 0, \ i = 1 \ldots n \tag{2}
\]
\[
\alpha_i^* g_i(w^*) = 0, \ i = 1 \ldots k \tag{3}
\]
\[
g_i(w^*) \leq 0, \ i = 1 \ldots k \tag{4}
\]
\[
\alpha_i^* \geq 0, \ i = 1 \ldots k \tag{5}
\]
Back to maximum margin perceptron

- We wanted to solve (rewritten slightly):
  \[
  \min \frac{1}{2} \|w\|^2 \\
  \text{w.r.t. } w, w_0 \\
  \text{s.t. } 1 - y_i(w \cdot x_i + w_0) \leq 0
  \]
- The Lagrangian is:
  \[
  L(w, w_0, \alpha) = \frac{1}{2} \|w\|^2 + \sum_i \alpha_i (1 - y_i(w \cdot x_i + w_0))
  \]
- The primal problem is: \( \min_{w, w_0} \max_{\alpha: \alpha_i \geq 0} L(w, w_0, \alpha) \)
- We will solve the dual problem: \( \max_{\alpha: \alpha_i \geq 0} \min_{w, w_0} L(w, w_0, \alpha) \)
- In this case, the optimal solutions coincide, because we have a quadratic objective and linear constraints (both of which are convex).
Solving the dual

- From KKT (2), the derivatives of $L(w, w_0, \alpha)$ wrt $w, w_0$ should be 0
- The condition on the derivative wrt $w_0$ gives $\sum_i \alpha_i y_i = 0$
- The condition on the derivative wrt $w$ gives:

$$w = \sum_i \alpha_i y_i x_i$$

$\Rightarrow$ Just like for the perceptron with zero initial weights, the optimal solution for $w$ is a linear combination of the $x_i$, and likewise for $w_0$.

- The output is

$$h_{w, w_0}(x) = \text{sign} \left( \sum_{i=1}^{m} \alpha_i y_i (x_i \cdot x) + w_0 \right)$$

$\Rightarrow$ Output depends on weighted dot product of input vector with training examples
Solving the dual

- By plugging these back into the expression for $L$, we get:

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

with constraints: $\alpha_i \geq 0$ and $\sum_{i} \alpha_i y_i = 0$
The support vectors

- Suppose we find optimal $\alpha$s (e.g., using a standard QP package)
- The $\alpha_i$ will be $> 0$ only for the points for which $1 - y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) = 0$
- These are the points lying on the edge of the margin, and they are called support vectors, because they define the decision boundary
- The output of the classifier for query point $\mathbf{x}$ is computed as:

$$\text{sgn} \left( \sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + w_0 \right)$$

Hence, the output is determined by computing the dot product of the point with the support vectors!
Example

Support vectors are in bold
Non-linearly separable data

- A linear boundary might be too simple to capture the class structure.
- One way of getting a nonlinear decision boundary in the input space is to find a linear decision boundary in an expanded space.

Thus, $x_i$ is replaced by $\phi(x_i)$, where $\phi$ is called a **feature mapping**
Non-linear SVMs: Feature Space

$$\Phi: x \rightarrow \varphi(x)$$
Non-linear SVMs: Feature Space
Margin optimization in feature space

- Replacing $x_i$ with $\phi(x_i)$, the optimization problem to find $w$ and $w_0$ becomes:
  \[ \min_{w, w_0} \|w\|^2 \]
  s.t. $y_i(w \cdot \phi(x_i) + w_0) \geq 1$

- Dual form:
  \[ \max_{\alpha_i} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \phi(x_i) \cdot \phi(x_j) \]
  w.r.t. $\alpha_i$
  s.t. $0 \leq \alpha_i$
  \[ \sum_{i=1}^{m} \alpha_i y_i = 0 \]
Feature space solution

- The optimal weights, in the expanded feature space, are \( w = \sum_{i=1}^{m} \alpha_i y_i \phi(x_i) \).

- Classification of an input \( x \) is given by:

\[
h_{w,w_0}(x) = \text{sign} \left( \sum_{i=1}^{m} \alpha_i y_i \phi(x_i) \cdot \phi(x) + w_0 \right)
\]

⇒ 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 $K : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ which corresponds to a dot product for some feature mapping $\phi$:

$$K(x_1, x_2) = \phi(x_1) \cdot \phi(x_2) \text{ for some } \phi.$$ 

- 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.
The “kernel trick”

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

- That is, we can solve the dual for the \( \alpha_i \):

\[
\begin{align*}
\max_{\alpha_i} & \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\
\text{w.r.t.} & \quad \alpha_i \\
\text{s.t.} & \quad 0 \leq \alpha_i \\
& \quad \sum_{i=1}^{m} \alpha_i y_i = 0
\end{align*}
\]

- The class of a new input \( \mathbf{x} \) is computed as:

\[
h_{\mathbf{w}, w_0}(\mathbf{x}) = \text{sign} \left( \sum_{i=1}^{m} \alpha_i y_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) + w_0 \right) = \text{sign} \left( \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + w_0 \right)
\]

- Often, \( K(\cdot, \cdot) \) can be evaluated in \( O(n) \) time—a big savings!
Nonlinear SVMs: The Kernel Trick

- An example:

2-dimensional vectors \( \mathbf{x} = [x_1 \ x_2] \);

let \( K(u,v) = (1 + u^T v)^2 \),

Need to show that \( K(u,v) = \phi(u)^T \phi(v) \):

\[
K(u,v) = (1 + u^T v)^2, \\
= 1 + u_1^2 v_1^2 + 2 u_1 v_1 u_2 v_2 + u_2^2 v_2^2 + 2 u_1 v_1 + 2 u_2 v_2 \\
= [1 \ u_1^2 \ \sqrt{2} u_1 u_2 \ u_2^2 \ \sqrt{2} u_1 \ \sqrt{2} u_2]^T [1 \ v_1^2 \ \sqrt{2} v_1 v_2 \ v_2^2 \ \sqrt{2} v_1 \ \sqrt{2} v_2] \\
= \phi(u)^T \phi(v), \text{ where } \phi(x) = [1 \ x_1^2 \ \sqrt{2} x_1 x_2 \ x_2^2 \ \sqrt{2} x_1 \ \sqrt{2} x_2]
\]
Nonlinear SVMs: The Kernel Trick

- Examples of commonly-used kernel functions:
  - Linear kernel: \( K(x_i, x_j) = x_i^T x_j \)
  - Polynomial kernel: \( K(x_i, x_j) = (1 + x_i^T x_j)^p \)
  - Gaussian (Radial-Basis Function (RBF)) kernel:
    \[
    K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)
    \]
  - Sigmoid:
    \[
    K(x_i, x_j) = \tanh(\beta_0 x_i^T x_j + \beta_1)
    \]

- In general, functions that satisfy Mercer’s condition can be kernel functions: Kernel matrix should be positive semidefinite.
Example

**Solutions:**
1) Nonlinear classifiers
2) Increase **dimensionality** of dataset and add a non-linear mapping $\Phi$

\[
\begin{bmatrix} x \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ x^2 \end{bmatrix}
\]

$2x^2 = 1 \iff 2x_2 = 1$
Example: String kernel

- Very important for DNA matching, text classification, ...
- Example: in DNA matching, we use a sliding window of length $k$ over the two strings that we want to compare
- The window is of a given size, and inside we can do various 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
- How do we prove this is a kernel?
Regularization with SVMs

- Kernels are a powerful tool for allowing non-linear, complex functions
- But now the number of parameters can be as high as the number of instances!
- With a very specific, non-linear kernel, each data point may be in its own partition
- With linear and logistic regression, we used regularization to avoid overfitting
- We need a method for allowing regularization with SVMs as well.
Soft margin linear classifier

- For the data that is not linear separable (noisy data, outliers, etc.)

- Slack variables $\xi_i$ can be added to allow misclassification of difficult or noisy data points
Soft margin classifiers

- Recall that in the linearly separable case, we compute the solution to the following optimization problem:
  \[
  \begin{align*}
  \text{min} & \quad \frac{1}{2} \| \mathbf{w} \|^2 \\
  \text{w.r.t.} & \quad \mathbf{w}, w_0 \\
  \text{s.t.} & \quad y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1
  \end{align*}
  \]

- If we want to allow misclassifications, we can relax the constraints to:
  \[
  y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1 - \xi_i
  \]

- If \( \xi_i \in (0, 1) \), the data point is within the margin
- If \( \xi_i \geq 1 \), then the data point is misclassified
- We define the soft error as \( \sum_i \xi_i \)
- We will have to change the criterion to reflect the soft errors
New problem formulation with soft errors

- Instead of:
  \[
  \min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|^2 \\
  \text{w.r.t. } \mathbf{w}, w_0 \\
  \text{s.t. } y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1
  \]
  
  we want to solve:
  \[
  \min_{\mathbf{w}, w_0, \xi_i} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i \\
  \text{w.r.t. } \mathbf{w}, w_0, \xi_i \\
  \text{s.t. } y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1 - \xi_i, \xi_i \geq 0
  \]

- Note that soft errors include points that are misclassified, as well as points within the margin

- There is a linear penalty for both categories

- The choice of the constant $C$ controls overfitting
A built-in overfitting framework

\[
\begin{align*}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_i \xi_i \\
\text{w.r.t.} \quad & \mathbf{w}, w_0, \xi_i \\
\text{s.t.} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1 - \xi_i \\
& \xi_i \geq 0
\end{align*}
\]

- If \( C \) is 0, there is no penalty for soft errors, so the focus is on maximizing the margin, even if this means more mistakes.
- If \( C \) is very large, the emphasis on the soft errors will cause decreasing the margin, if this helps to classify more examples correctly.
- Internal cross-validation is a good way to choose \( C \) appropriately.
Like before, we can write a Lagrangian for the problem and then use the dual formulation to find the optimal parameters:

\[
L(w, w_0, \alpha, \xi, \mu) = \frac{1}{2}||w||^2 + C \sum_i \xi_i \\
+ \sum_i \alpha_i (1 - \xi_i - y_i(w_i \cdot x_i + w_0)) + \sum_i \mu_i \xi_i
\]

All the previously described machinery can be used to solve this problem.

Note that in addition to \(\alpha_i\), we have coefficients \(\mu_i\), which ensure that the errors are positive, but do not participate in the decision boundary.

\[
\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\phi(x_i) \cdot \phi(x_j))
\]

with constraints: \(0 \leq \alpha_i \leq C\) and \(\sum_i \alpha_i y_i = 0\).
Soft margin optimization with kernels

- Replacing $x_i$ with $\phi(x_i)$, the optimization problem to find $w$ and $w_0$ becomes:
  \[
  \min_{w, w_0, \zeta_i} \|w\|^2 + C \sum_i \zeta_i \\
  \text{s.t. } y_i(w \cdot \phi(x_i) + w_0) \geq (1 - \zeta_i) \\
  \zeta_i \geq 0
  \]

- Dual form and solution have similar forms to what we described last time, but in terms of kernels
Getting SVMs to work in practice

- Two important choices:
  - Kernel (and kernel parameters)
  - Regularization parameter $C$

- The parameters may interact!
  E.g. for Gaussian kernel, the larger the width of the kernel, the more biased the classifier, so low $C$ is better

- Together, these control overfitting: always do an internal parameter search, using a validation set!

- Overfitting symptoms:
  - Low margin
  - Large fraction of instances are support vectors
Solving the quadratic optimization problem

- Many approaches exist
- Because we have constraints, gradient descent does not apply directly (the optimum might be outside of the feasible region)
- Platt’s algorithm is the fastest current approach, based on coordinate ascent
Coordinate ascent

- Suppose you want to find the maximum of some function $F(\alpha_1, \ldots, \alpha_n)$
- Coordinate ascent optimizes the function by repeatedly picking an $\alpha_i$ and optimizing it, while all other parameters are fixed
- There are different ways of looping through the parameters:
  - Round-robin
  - Repeatedly pick a parameter at random
  - Choose next the variable expected to make the largest improvement
  - ...
Our optimization problem (dual form)

\[
\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\phi(x_i) \cdot \phi(x_j))
\]

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

- Suppose we want to optimize for \(\alpha_1\) while \(\alpha_2, \ldots, \alpha_n\) are fixed
- We cannot do it because \(\alpha_1\) will be completely determined by the last constraint: \(\alpha_1 = -y_1 \sum_{i=2}^{m} \alpha_i y_i\)
- Instead, we have to optimize \textit{pairs of parameters} \(\alpha_i, \alpha_j\) together
Sequential minimal optimization (SMO)

- Suppose that we want to optimize $\alpha_1$ and $\alpha_2$ together, while all other parameters are fixed.
- We know that $y_1 \alpha_1 + y_2 \alpha_2 = -\sum_{i=1}^{m} y_i \alpha_i = \xi$, where $\xi$ is a constant.
- So $\alpha_1 = y_1 (\xi - y_2 \alpha_2)$ (because $y_1$ is either $+1$ or $-1$ so $y_1^2 = 1$).
- This defines a line, and any pair $\alpha_1, \alpha_2$ which is a solution has to be on the line.
- We also know that $0 \leq \alpha_1 \leq C$ and $0 \leq \alpha_2 \leq C$, so the solution has to be on the line segment inside the rectangle below.
Sequential minimal optimization (SMO)

- By plugging $\alpha_1$ back in the optimization criterion, we obtain a quadratic function of $\alpha_2$, whose optimum we can find exactly.
- If the optimum is inside the rectangle, we take it.
- If not, we pick the closest intersection point of the line and the rectangle.
- This procedure is very fast because all these are simple computations.
Multi-class classification

- one-vs-all
- \( n \) classifiers
- choose the class with the largest margin

- one-vs-one
- \( \frac{n(n-1)}{2} \) classifiers
- choose the class chosen by most classifiers
Complexity

- Quadratic programming is expensive in the number of training examples
- Platt’s SMO algorithm is quite fast though, and other fancy optimization approaches are available
- Best packages can handle 50,000+ instances, but not more than 100,000
- On the other hand, number of attributes can be very high (strength compared to neural nets)
- Evaluating a SVM is *slow if there are a lot of support vectors.*
- Dictionary methods attempt to select a subset of the data on which to train.