Support Vector Machines

In SVMs we are trying to find a decision boundary that maximizes the "margin" or the "width of the road" separating the positives from the negative training data points.

To find this we **minimize:** $\frac{1}{2}\|\vec{w}\|^2$ subject to the constraints $y_i(\vec{w}\cdot\vec{x_i}+b)\geq 1$

The resulting Lagrange multiplier equation we try to optimize is:

$$L=\frac{1}{2}\|w\|^2-\sum_i\alpha_i(y_i(\vec{w}\cdot\vec{x_i}+b)-1)$$

Solving the above Lagrangian optimization problem will give us w, b, and alphas, parameters that determines a **unique** maximal margin (road) solution. On the maximum margin "road", the +ve, and -ve points that stride the "gutter" lines are called ***support vectors***. The decision boundary lies at the middle of the road. The definition of the "road" is dependent only on the support vectors, so changing (adding deleting) non-support vector points will not change the solution. Note, that widest "road" is a 2D concept. If the problem is in 3D we want the widest region bounded by two planes; in even higher dimensions, a subspace bounded by two hyperplanes.

Solving for the Lagrange multiplier α_is in general requires numerical optimization methods that are beyond the scope of this class. In practice, you use Quadratic Programming solvers. A popular algorithm for solving SVMs is Platt's SMO (Sequential Minimal Optimization) algorithm. For SVM problems on quizzes, we generally just ask you to solve for the values of w, b and alphas using algebra and/or geometry.

Useful Equations for solving SVM questions

A. Equations derived from optimizing the Lagrangian:

1. **Partial of the Lagrangian wrt to b**: From $\frac{\partial L}{\partial b}=0$

$\sum_i\alpha_i y_i=0$	Note that $y_i\in\{-1,+1\}$ and $\alpha_i=0$ for non-support vectors.

Sum of all alphas (support vector weights) with their signs should add to 0.

2. **Partial of the Lagrangian wrt to w**: From $\frac{\partial L}{\partial w}=0$

$\sum_i\alpha_i y_i\vec{x_i}=\vec{w}$	For when using a linear kernel. The summation only contains support vectors. Support vectors are training data points with $\alpha_i>0$
$\sum_i\alpha_i y_i\phi(\vec{x_i})=\vec{w}$	For when using a decomposable kernel (see definition below).

Sum of alphas, ys of support vectors wrt to vector w.

B. Equations from the boundaries and constraints:

3. **The Decision boundary**:

$h(\vec{x}) = \sum_i \alpha_i y_i K(\vec{x_i}, \vec{x}) + b \geq 0$	General form, for any kernel. To classify an unknown $\vec{x}$, we compute the kernel function $K(\vec{x_i}, \vec{x})$ against each of the **support vectors** $\vec{x_i}$. Support vectors are training data points with $\alpha_i > 0$
$h(\vec{x}) = \sum_i [(\alpha_i y_i \vec{x_i}) \cdot \vec{x}] + b \geq 0$ $h(\vec{x}) = \vec{w} \cdot \vec{x} + b \geq 0$	For when using a linear kernel $K(\vec{x_i}, \vec{x}) = \vec{x_i} \cdot \vec{x}$

4. **Positive gutter:**

$h(\vec{x}) = \sum_i \alpha_i y_i K(\vec{x_i}, \vec{x}) + b = 1$	General form, for any kernel.
$h(\vec{x}) = \sum_i [(\alpha_i y_i \vec{x_i}) \cdot \vec{x}] + b = 1$ $h(\vec{x}) = \vec{w} \cdot \vec{x} + b = 1$	For use when the Kernel is linear.

5. **Negative gutter:**

$h(\vec{x}) = \sum_i \alpha_i y_i K(\vec{x_i}, \vec{x}) + b = -1$	$h(\vec{x}) = \vec{w} \cdot \vec{x} + b = -1$

6. **The width of the margin (or road):**

$$\text{width of road} \equiv m = \frac{2}{\|\vec{w}\|} \qquad \text{where,} \qquad \|\vec{w}\| = \sqrt{\sum_i w_i^2}$$

Alternate formula for the two support vector case:

$$\text{width of road} \equiv m = \frac{\vec{w}}{\|\vec{w}\|} \cdot (\vec{x}_+ - \vec{x}_-)$$

*This equation is useful when solving SVM problems in 1D or 2D, where the width of the road can be **visually determined**.*

Common SVM Kernels:

Linear Kernel	$K(\vec{u}, \vec{v}) = \vec{u} \cdot \vec{v}$ In document classification, feature vectors are composed of binary word features: I(word=foo) outputs 1 if the word "foo" appears in the document 0 if it does not. Each document is represented as |vocabulary| length feature vectors. Support vectors found are generally particularly salient documents (documents best at discriminating topics being classified).

<table>
<tr>
<td>Decomposable Kernels

Idea: Define $\phi(\vec{u})$ that transforms input vectors into a different (usually higher) dimensional space where the data is (more easily) linearly separable.</td>
<td>$K(\vec{u},\vec{v}) = \phi(\vec{u}) \cdot \phi(\vec{v})$

Example:
$\phi(\vec{u}) = \begin{bmatrix} \cos(u_1) \\ \sin(u_2) \end{bmatrix}$ $K(\vec{u},\vec{v}) = \cos(\vec{u_1})\cos(\vec{v_1}) + \sin(\vec{u_2})\sin(\vec{v_2})$</td>
</tr>
<tr>
<td>Polynomial Kernel</td>
<td>$K(\vec{u},\vec{v}) = (\vec{u} \cdot \vec{v} + b)^n$ n > 1

Example: Quadratic Kernel: $K(\vec{u},\vec{v}) = (\vec{u} \cdot \vec{v} + b)^2$
• In 2D resulting decision boundary can look parabolic, linear or hyperbolic depending on which terms in the expansion dominate.
• Here is an expansion of the quadratic kernel, with u = [x, y]
$K(\vec{u},\vec{v}) = \left(\begin{bmatrix} x \\ y \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + b \right)^2$
$= (v_1 x + v_2 y + b)^2$
$= [(v_1^2)x^2 + (v_2^2)y^2] + [b^2 + (2v_1 b)x + (2v_2 b)y] + [(2v_1 v_2)xy]$

HW: Try this Kernel using Professor Winston's demo</td>
</tr>
<tr>
<td>Radial Basis Function (RBF) or Gaussian Kernel

• Will fit almost any data. May exhibit overfitting when used improperly.
• Similar to KNN but with all points having a vote; weight of each vote determined by Gaussian
◦ Points farther away get less of a vote than points</td>
<td>$K(\vec{u},\vec{v}) = \exp\left(-\frac{\|\vec{u} - \vec{v}\|^2}{2\sigma^2} \right)$
In 2D generated decision boundaries resemble contour circles around clusters of +ve and -ve points. Support vectors are generally +ve or -ve points that are closest to the opposing cluster. The contour space drawn results from sum of support vector Gaussians.

HW: Try this Kernel using Professor Winston's demo

When σ^2 is large you get flatter Gaussians. When σ^2 is small you get sharper Gaussians. (Hence when using a small σ^2 contour density will appear closer / denser around support vector points).

Here is the Kernel in-2D expanded out, with u = [x, y]
$K(\vec{u},\vec{v}) = \exp\left(-\frac{(x - v_1)^2 + (y - v_2)^2}{2\sigma^2} \right)$
As a point gets closer to a support vector it approaches exp(0) = 1. As a point moves far away from a support vector it approaches exp(-infinity) = 0.</td>
</tr>
</table>