Kernel Methods and Support Vector Machines

Oliver Schulte - CMPT 726

Bishop PRML Ch. 6

Support Vector Machines

Defining Characteristics

- Like logistic regression, good for continuous input features, discrete target variable.
- Like nearest neighbor, a *kernel method*: classification is based on weighted similar instances. The kernel defines similarity measure.
- Sparsity: Tries to find a few important instances, the *support vectors*.
- Intuition: Netflix recommendation system.

SVMs: Pros and Cons

Pros

- Very good classification performance, basically unbeatable.
- Fast and scaleable learning.
- Pretty fast inference.

Cons

- No model is built, therefore black-box.
- Still need to specify kernel function (like specifying basis functions).
- Issues with multiple classes, can use probabilistic version. (Relevance Vector Machine).

Two Views of SVMs

Theoretical View: linear separator

- SVM looks for linear separator but in *new feature space*.
- Uses a new criterion to choose a line separating classes: *max-margin*.

User View: kernel-based classification

- User specifies a kernel function.
- SVM learns weights for instances.
- Classification is performed by taking average of the labels of other instances, weighted by a) similarity b) instance weight.

Nice demo on web
`http://www.youtube.com/watch?v=3liCbRZPrZA`.

Two Views of SVMs

Theoretical View: linear separator

- SVM looks for linear separator but in *new feature space*.
- Uses a new criterion to choose a line separating classes: *max-margin*.

User View: kernel-based classification

- User specifies a kernel function.
- SVM learns weights for instances.
- Classification is performed by taking average of the labels of other instances, weighted by a) similarity b) instance weight.

Nice demo on web
http://www.youtube.com/watch?v=3liCbRZPrZA.

Two Views of SVMs

Theoretical View: linear separator

- SVM looks for linear separator but in *new feature space*.
- Uses a new criterion to choose a line separating classes: *max-margin*.

User View: kernel-based classification

- User specifies a kernel function.
- SVM learns weights for instances.
- Classification is performed by taking average of the labels of other instances, weighted by a) similarity b) instance weight.

Nice demo on web

`http://www.youtube.com/watch?v=3liCbRZPrZA`.

Two Views of SVMs

Theoretical View: linear separator

- SVM looks for linear separator but in *new feature space*.
- Uses a new criterion to choose a line separating classes: *max-margin*.

User View: kernel-based classification

- User specifies a kernel function.
- SVM learns weights for instances.
- Classification is performed by taking average of the labels of other instances, weighted by a) similarity b) instance weight.

Nice demo on web
`http://www.youtube.com/watch?v=3liCbRZPrZA`.

Outline

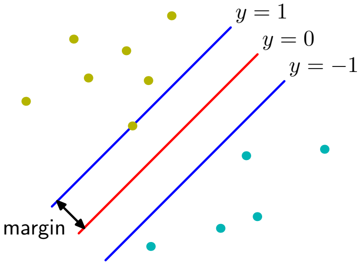
Linear Classification Revisited

- Consider a two class classification problem
- Use a linear model

  $$y(\boldsymbol{x}) = \boldsymbol{w} \bullet \boldsymbol{x} + b$$

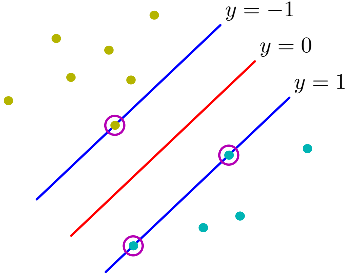
  followed by a threshold function
- For now, let's assume training data are linearly separable (possibly after mapping to higher-dimensional space).
  - Recall that the perceptron would converge to a perfect classifier for such data
  - But there are many such perfect classifiers

Max Margin Classifiers



- We can define the margin of a classifier as the minimum distance to any example
- In support vector machines the decision boundary which maximizes the margin is chosen.
- Intuitively, this is the line “right in the middle” between the two classes.

Support Vectors



- The support vectors are the points at minimum distance to the decision boundary.
- The max-margin boundary depends only on the support vectors: other data points do not matter for classification and need not be stored.

Outline

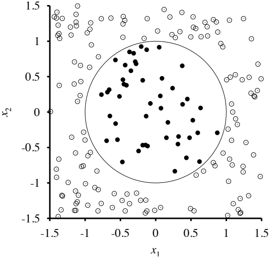
Example: X-OR

- X-OR problem: class of (x_1, x_2) is positive iff $x_1 \cdot x_2 > 0$.
- Not linearly separable in input space, but linearly separable if *we add extra dimensions.*
- Use 6 basis functions $\phi(x_1, x_2) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$.
- Simple classifier $y(x_1, x_2) = \phi_5(x_1, x_2) = \sqrt{2}x_1x_2$.
  - *Linear* in basis function space.

Example: X-OR

- X-OR problem: class of (x_1, x_2) is positive iff $x_1 \cdot x_2 > 0$.
- Not linearly separable in input space, but linearly separable if *we add extra dimensions.*
- Use 6 basis functions $\phi(x_1, x_2) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$.
- Simple classifier $y(x_1, x_2) = \phi_5(x_1, x_2) = \sqrt{2}x_1x_2$.
  - *Linear* in basis function space.

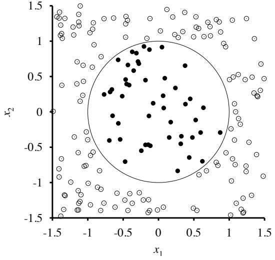
Linear Separability Example



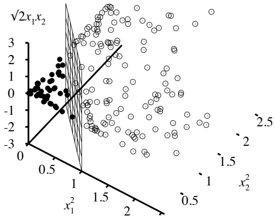
Decision Boundary $x_1^2 + x_2^2 \leq 1$

3-D mapping $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$

Linear Separability Example



Decision Boundary $x_1^2 + x_2^2 \leq 1$



3-D mapping $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$.

The Kernel Trick

- There can be many extra dimensions, even infinite (see assignment).
  - Don't want to compute basis function mapping $\phi(\boldsymbol{x})$.
- Key insight 1: Linear classification requires *only the dot product*.
- Key insight 2: The high-dimensional dot product $\phi(\boldsymbol{x}) \bullet \phi(z)$ can often be computed as a kernel function of the input vectors only:

$$\phi(\boldsymbol{x}) \bullet \phi(z) = k(\boldsymbol{x}, z).$$

Kernel Trick Example

- Consider again the X-OR 6 basis functions $\phi(x_1,x_2) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$.
- Exercise: find a closed form expression for $\phi(\boldsymbol{x}) \bullet \phi(\boldsymbol{z})$
- Solution: Dot product $\phi(\boldsymbol{x}) \bullet \phi(\boldsymbol{z}) = (1 + \boldsymbol{x} \bullet \boldsymbol{z})^2 = k(\boldsymbol{x}, \boldsymbol{z})$.
- A quadratic kernel.

Kernel Trick Example

- Consider again the X-OR 6 basis functions $\phi(x_1, x_2) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$.
- Exercise: find a closed form expression for $\phi(\boldsymbol{x}) \bullet \phi(\boldsymbol{z})$
- Solution: Dot product $\phi(\boldsymbol{x}) \bullet \phi(\boldsymbol{z}) = (1 + \boldsymbol{x} \bullet \boldsymbol{z})^2 = k(\boldsymbol{x}, \boldsymbol{z})$.
- A quadratic kernel.

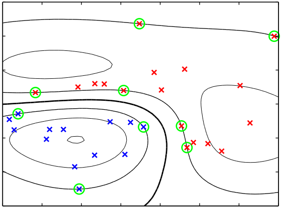
The Kernel Classification Formula

- Suppose we have a kernel function k and N labelled instances with weights $a_n \geq 0, n = 1, \ldots, N$.
- As with the perceptron, the target labels +1 are for positive class, -1 for negative class.
- Then

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b$$

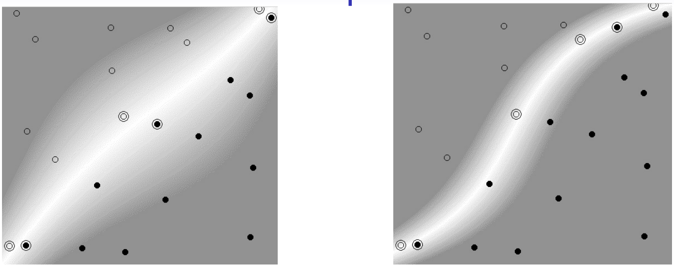
- $\boldsymbol{x}$ is classified as positive if $y(\boldsymbol{x}) > 0$, negative otherwise.
- If $a_n > 0$, then $\boldsymbol{x}_n$ is a support vector.
- Don't need to store other vectors.
- $\boldsymbol{a}$ will be sparse - many zeros.

Example1



- SVM with Gaussian kernel
- Support vectors circled.
- They are the closest to the other class.
- Note non-linear decision boundary in x space

Example2



- From Burges, *A Tutorial on Support Vector Machines for Pattern Recognition* (1998)
- SVM trained using cubic polynomial kernel
  $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1 \bullet \boldsymbol{x}_2 + 1)^3$
- Left is linearly separable
  - Note decision boundary is almost linear, even using cubic polynomial kernel
- Right is not linearly separable
  - But is separable using polynomial kernel

Learning the Instance Weights

- The max-margin classifier is found by solving the following problem:
- Maximize wrt $\boldsymbol{a}$

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

  subject to the constraints
  - $a_n \geq 0,\ n = 1, \ldots, N$
  - $\sum_{n=1}^{N} a_n t_n = 0$
- It is quadratic, with linear constraints, convex in $\boldsymbol{a}$
- Optimal $\boldsymbol{a}$ can be found
  - With large datasets, local search strategies employed

let's check SVM demo

http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml

Learning the Instance Weights

- The max-margin classifier is found by solving the following problem:
- Maximize wrt $\boldsymbol{a}$

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

  subject to the constraints
  - $a_n \geq 0,\ n = 1, \ldots, N$
  - $\sum_{n=1}^{N} a_n t_n = 0$
- It is quadratic, with linear constraints, convex in $\boldsymbol{a}$
- Optimal $\boldsymbol{a}$ can be found
  - With large datasets, local search strategies employed

let's check SVM demo

`http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml`

Outline

Valid Kernels

- Valid kernels: if $k(\cdot,\cdot)$ satisfies:
  - Symmetric; $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_j, \boldsymbol{x}_i)$
  - Positive definite; for any $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the Gram matrix $\boldsymbol{K}$ must be positive semi-definite:

$$\boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \ldots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \ldots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

  - Positive semi-definite means $\boldsymbol{x} \bullet \boldsymbol{K}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$ (like metric)

then $k(\cdot,\cdot)$ corresponds to a dot product in some space ϕ

  - a.k.a. Mercer kernel, admissible kernel, reproducing kernel
  - Theorem of Mercer's 1909!

Examples of Kernels

- Some kernels:
  - Linear kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1 \bullet \boldsymbol{x}_2$
  - Polynomial kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (1 + \boldsymbol{x}_1 \bullet \boldsymbol{x}_2)^d$
    - Contains all polynomial terms up to degree d
  - Gaussian kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2 / 2\sigma^2)$
    - Infinite dimension feature space

Constructing Kernels

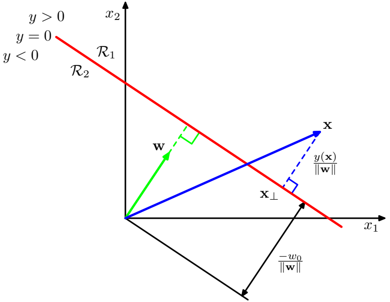
- Can build new valid kernels from existing valid ones:
  - $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = ck_1(\boldsymbol{x}_1, \boldsymbol{x}_2)$, $c > 0$
  - $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_1(\boldsymbol{x}_1, \boldsymbol{x}_2) + k_2(\boldsymbol{x}_1, \boldsymbol{x}_2)$
  - $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_1(\boldsymbol{x}_1, \boldsymbol{x}_2)k_2(\boldsymbol{x}_1, \boldsymbol{x}_2)$
  - $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(k_1(\boldsymbol{x}_1, \boldsymbol{x}_2))$
- Table on p. 296 gives many such rules

More Kernels

- Stationary kernels are only a function of the difference between arguments: $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\boldsymbol{x}_1 - \boldsymbol{x}_2)$
  - Translation invariant in input space: $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\boldsymbol{x}_1 + \boldsymbol{c}, \boldsymbol{x}_2 + \boldsymbol{c})$
- Homogeneous kernels, a. k. a. radial basis functions only a function of magnitude of difference: $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(||\boldsymbol{x}_1 - \boldsymbol{x}_2||)$
- Set subsets $k(A_1, A_2) = 2^{|A_1 \cap A_2|}$, where $|A|$ denotes number of elements in A
- Domain-specific: think hard about your problem, figure out what it means to be similar, define as $k(\cdot, \cdot)$, prove positive definite.

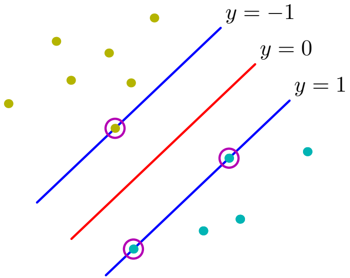
Outline

Marginal Geometry



- See assignment.
- Projection of $\boldsymbol{x}$ in $\boldsymbol{w}$ dir. is $\frac{\boldsymbol{w} \bullet \boldsymbol{x}}{||\boldsymbol{w}||}$
- $y(\boldsymbol{x}) = 0$ when $\boldsymbol{w} \bullet \boldsymbol{x} = -b$, or $\frac{\boldsymbol{w} \bullet \boldsymbol{x}}{||\boldsymbol{w}||} = \frac{-b}{||\boldsymbol{w}||}$
- So $\frac{\boldsymbol{w} \bullet \boldsymbol{x}}{||\boldsymbol{w}||} - \frac{-b}{||\boldsymbol{w}||} = \frac{y(\boldsymbol{x})}{||\boldsymbol{w}||}$ is signed distance to decision boundary

Support Vectors



- Assuming data are separated by the hyperplane, distance to decision boundary is $\frac{t_n y(\boldsymbol{x}_n)}{||\boldsymbol{w}||}$
- The maximum margin criterion chooses $\boldsymbol{w}, b$ by:

$$\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{||\boldsymbol{w}||} \min_n [t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n + b)] \right\}$$

- Points with this min value are known as support vectors

Canonical Representation

- This optimization problem is complex:

$$\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{||\boldsymbol{w}||} \min_n [t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n) + b)] \right\}$$

- Exercise: Prove that rescaling $w \to \kappa w$ and $b \to \kappa b$ does not change distance $\frac{t_n y(\boldsymbol{x}_n)}{||\boldsymbol{w}||}$ (many equiv. answers)
- So for $\boldsymbol{x}_*$ closest to surface, can set (how?):

$$t_*(\boldsymbol{w} \bullet \boldsymbol{x}_* + b) = 1$$

- All other points are at least this far away:

$$\forall n \ , \ t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n + b) \geq 1$$

- Knowing that *min* distance = 1, the optimization becomes:

$$\arg\max_{\boldsymbol{w},b} \frac{1}{||\boldsymbol{w}||} = \arg\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2$$

Canonical Representation

- This optimization problem is complex:

$$\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{||\boldsymbol{w}||} \min_n [t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n) + b)] \right\}$$

- Exercise: Prove that rescaling $\boldsymbol{w} \to \kappa \boldsymbol{w}$ and $b \to \kappa b$ does not change distance $\frac{t_n y(\boldsymbol{x}_n)}{||\boldsymbol{w}||}$ (many equiv. answers)
- So for x_* closest to surface, can set (how?):

$$t_*(w \bullet x_* + b) = 1$$

- All other points are at least this far away:

$$\forall n \ , \ t_n(w \bullet x_n + b) \geq 1$$

- Knowing that *min* distance = 1, the optimization becomes:

$$\arg\max_{w,b} \frac{1}{||w||} = \arg\min_{w,b} \frac{1}{2}||w||^2$$

Canonical Representation

- This optimization problem is complex:

$$\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{||\boldsymbol{w}||} \min_n [t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n) + b)] \right\}$$

- Exercise: Prove that rescaling $\boldsymbol{w} \to \kappa \boldsymbol{w}$ and $b \to \kappa b$ does not change distance $\frac{t_n y(\boldsymbol{x}_n)}{||\boldsymbol{w}||}$ (many equiv. answers)
- So for $\boldsymbol{x}_*$ closest to surface, can set (how?):

$$t_*(\boldsymbol{w} \bullet \boldsymbol{x}_* + b) = 1$$

- All other points are at least this far away:

$$\forall n \ , \ t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n + b) \geq 1$$

- Knowing that min distance = 1, the optimization becomes:

$$\arg\max_{\boldsymbol{w},b} \frac{1}{||\boldsymbol{w}||} = \arg\min_{\boldsymbol{w},b} \frac{1}{2} ||\boldsymbol{w}||^2$$

Canonical Representation

- This optimization problem is complex:

$$\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{||\boldsymbol{w}||} \min_{n} [t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n) + b)] \right\}$$

- Exercise: Prove that rescaling $\boldsymbol{w} \to \kappa \boldsymbol{w}$ and $b \to \kappa b$ does not change distance $\frac{t_n y(\boldsymbol{x}_n)}{||\boldsymbol{w}||}$ (many equiv. answers)
- So for $\boldsymbol{x}_*$ closest to surface, can set (how?):

$$t_*(\boldsymbol{w} \bullet \boldsymbol{x}_* + b) = 1$$

- All other points are at least this far away:

$$\forall n \ , \ t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n + b) \geq 1$$

- Knowing that min distance = 1, the optimization becomes:

$$\arg\max_{\boldsymbol{w},b} \frac{1}{||\boldsymbol{w}||} = \arg\min_{\boldsymbol{w},b} \frac{1}{2} ||\boldsymbol{w}||^2$$

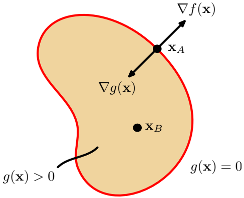
Canonical Representation

- So the optimization problem is now a constrained optimization problem:

$$\begin{aligned} & \arg\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2 \\ s.t. \quad & \forall n \ , \ t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n + b) \geq 1 \end{aligned}$$

- To solve this, we need to use Lagrange multipliers

Lagrange Multipliers - Inequality Constraints



Consider the problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$s.t. \quad g_1(\boldsymbol{x}) \geq 0, \ldots, g_N(\boldsymbol{x}) \geq 0$$

Solutions are stationary points of the Lagrangian

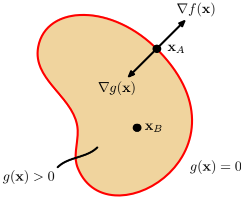
$$L(\boldsymbol{x}, \boldsymbol{a}) = f(\boldsymbol{x}) - \sum_{n=1}^{N} a_n g_n(\boldsymbol{x})$$

where for each n we have the KKT conditions,

- $a_n \geq 0$
- $g_n(\boldsymbol{x}) \geq 0$
- $a_n g_n(\boldsymbol{x}) = 0$

Therefore $a_n = 0$ (inactive constraint) or $g_n(\boldsymbol{x}) = 0$ (active constraint).

Lagrange Multipliers - Inequality Constraints



Consider the problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$s.t. \qquad g_1(\boldsymbol{x}) \geq 0, \ldots, g_N(\boldsymbol{x}) \geq 0$$

Solutions are stationary points of the Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{a}) = f(\boldsymbol{x}) - \sum_{n=1}^{N} a_n g_n(\boldsymbol{x})$$

where for each n we have the KKT conditions,

- $a_n \geq 0$
- $g_n(\boldsymbol{x}) \geq 0$
- $a_n g_n(\boldsymbol{x}) = 0$

Therefore $a_n = 0$ (inactive constraint) or $g_n(\boldsymbol{x}) = 0$ (active constraint).

Now Where Were We

- So the optimization problem is now a constrained optimization problem:

$$\begin{aligned} & \arg\min_{\boldsymbol{w},b} \frac{||\boldsymbol{w}||^2}{2} \\ s.t. \qquad & \forall n \ , \ t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n + b) \geq 1 \end{aligned}$$

- For this problem, the Lagrangian (with N multipliers a_n) is:

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{||\boldsymbol{w}||^2}{2} - \sum_{n=1}^{N} a_n \left\{ t_n(\boldsymbol{w} \bullet \boldsymbol{x}_n + b) - 1 \right\}$$

- We can find the derivatives of L wrt $\boldsymbol{w}, b$ and set to 0:

$$\begin{aligned} \boldsymbol{w} &= \sum_{n=1}^{N} a_n t_n \boldsymbol{x}_n \\ 0 &= \sum_{n=1}^{N} a_n t_n \end{aligned}$$

The Dual Formulation

- Recall the condition:

$$\boldsymbol{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{x}_n$$

- Exercise: Show that $\frac{||\boldsymbol{w}||^2}{2} = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (\boldsymbol{x}_n \bullet \boldsymbol{x}_m)$.
- Exercise: Show that

$$\sum_{n=1}^{N} a_n \left\{ t_n (\boldsymbol{w} \bullet \boldsymbol{x}_n + b) - 1 \right\} =$$

$$\sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (\boldsymbol{x}_n \bullet \boldsymbol{x}_m) + b \sum_{n=1}^{N} a_n t_n - \sum_{n=1}^{N} a_n$$

Solving The Dual Formulation

- Combining the exercise results, we obtain the dual Lagrangian as a function of the Lagrange multipliers only:

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (\boldsymbol{x}_n \bullet \boldsymbol{x}_m)$$

- The stationary points of $\tilde{L}$ provide lower bounds on the original problem, so we want to maximize $\tilde{L}$ (see `http://en.wikipedia.org/wiki/Lagrange_duality`).
- Apply the kernel trick to replace $\bullet$ with kernel k, and remember the constraints on the Lagrange multipliers, to arrive at the following problem:
- Maximize $\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ subject to the constraints that
  - $a_n \geq 0$, $n = 1, \ldots, N$
  - $\sum_{n=1}^{N} a_n t_n = 0$

From The Dual Solution $\boldsymbol{a}$ to a Classifier

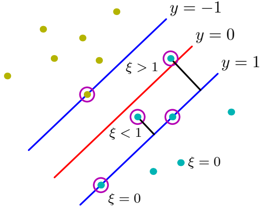
- Given the solution $\boldsymbol{a}$, we have formulas for $\boldsymbol{w}$ and for b (omitted). Then classify as follows:

$$
\begin{aligned}
\boldsymbol{w} &= \sum_{n=1}^{N} a_n t_n \phi(\boldsymbol{x}_n) \\
y(\boldsymbol{x}) &= \boldsymbol{w} \bullet \phi(\boldsymbol{x}) + b = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b
\end{aligned}
$$

- Recall that every constraint is either inactive ($a_n = 0$) or active ($a_n > 0$ and $t_n y(\boldsymbol{x}_n) = 1$).
- If $a_n > 0$, then $\boldsymbol{x}_n$ is a support vector.
- $\boldsymbol{a}$ will be sparse - many zeros.
  - Don't need to store $\boldsymbol{x}_n$ for which $a_n = 0$

Outline

Non-Separable Data

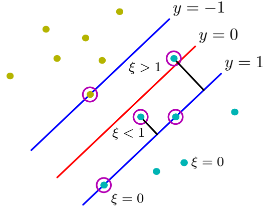


- For most problems, data will not be linearly separable (even in feature space ϕ)
- Can relax the constraints from

$$t_n y(\boldsymbol{x}_n) \geq 1 \quad \text{to} \quad t_n y(\boldsymbol{x}_n) \geq 1 - \xi_n$$

- The $\xi_n \geq 0$ are called slack variables
  - $\xi_n = 0$, satisfy original problem, so x_n is on margin or correct side of margin
  - $0 < \xi_n < 1$, inside margin, but still correctly classifed
  - $\xi_n > 1$, mis-classified

Loss Function For Non-separable Data



- Non-zero slack variables are bad, penalize while maximizing the margin:

$$\min C \sum_{n=1}^{N} \xi_n + \frac{1}{2}||\boldsymbol{w}||^2$$

- Constant $C > 0$ controls importance of large margin versus incorrect (non-zero slack)
  - Set using cross-validation
- Optimization is same quadratic, different constraints, convex

SVM Loss Function

- The SVM for the separable case solved the problem:

$$\arg\min_{\boldsymbol{w}} \frac{1}{2}||\boldsymbol{w}||^2$$
$$s.t. \qquad \forall n\ ,\ t_n y_n \geq 1$$

- Can write this as:

$$\arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} E_\infty(t_n y_n) + \lambda ||\boldsymbol{w}||^2$$

where $E_\infty(z) = 0$ if $z \geq 1$, ∞ otherwise

- Non-separable case relaxes this to be:

$$\arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} E_{SV}(t_n y_n) + \lambda ||\boldsymbol{w}||^2$$

where $E_{SV}(z) = [z]_+$ hinge loss
  - $[z]_+ = z$ if $u \leq 1$, 0 otherwise

SVM Loss Function

- The SVM for the separable case solved the problem:

$$\begin{array}{ll} & \arg\min_{\boldsymbol{w}} \frac{1}{2}||\boldsymbol{w}||^2 \\ s.t. & \forall n \ , \ t_n y_n \geq 1 \end{array}$$

- Can write this as:

$$\arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} E_\infty(t_n y_n) + \lambda ||\boldsymbol{w}||^2$$

where $E_\infty(z) = 0$ if $z \geq 1$, ∞ otherwise

- Non-separable case relaxes this to be:

$$\arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} E_{SV}(t_n y_n) + \lambda ||\boldsymbol{w}||^2$$

where $E_{SV}(z) = [z]_+$ hinge loss
  - $[z]_+ = z$ if $u \leq 1$, 0 otherwise

SVM Loss Function

- The SVM for the separable case solved the problem:

$$\begin{aligned} & \arg\min_{\boldsymbol{w}} \frac{1}{2}||\boldsymbol{w}||^2 \\ s.t. \quad & \forall n \text{ , } t_n y_n \geq 1 \end{aligned}$$

- Can write this as:

$$\arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} E_\infty(t_n y_n) + \lambda||\boldsymbol{w}||^2$$

where $E_\infty(z) = 0$ if $z \geq 1$, ∞ otherwise

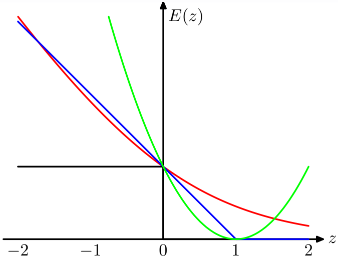
- Non-separable case relaxes this to be:

$$\arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} E_{SV}(t_n y_n) + \lambda||\boldsymbol{w}||^2$$

where $E_{SV}(z) = [z]_+$ hinge loss

  - $[z]_+ = z$ if $u \leq 1$, 0 otherwise

Loss Functions



- Linear classifiers, compare loss function used for learning
- $z = y_n t_n \leq 0$ iff there is an error (with $t_n \in \{+1, -1\}$).
- $z = y_n t_n \geq 1$ iff the point is on the right side of the margin boundary.
    - Black is misclassification error
    - Transformed simple linear classifier, squared error: $(y_n - t_n)^2$
    - Transformed logistic regression, cross-entropy error: $t_n \ln y_n$
    - SVM, hinge loss:
        - positive *only* if the point is on the wrong side of the margin boundary. Sparse solutions.

Two Views of Learning as Optimization

- The original SVM goal was of the form:
  - Find the simplest hypothesis that is consistent with the data, or
  - Maximize simplicity, given a consistency constraint.
- This general idea appears in much scientific model building, in image processing, and other applications.
- Bayesian methods use a criterion of the form
  - Find a trade-off between simplicity and data fit, or
  - Maximize sum of the type (data fit - λ simplicity)
  - e.g., $ln(P(D|M)) - \lambda ln(P(M))$ where the model prior M is higher for simpler models.

Pros and Cons of Learning Criteria

- The Bayesian approach has a solid probabilistic foundation in Bayes' theorem.
- Seems to be especially suitable for noisy data.
- The constraint-based approach is often easy for users to understand.
- Often leads to sparser simpler models.
- Suitable for "clean" data.

Conclusion

- Readings: Ch. 6.1-6.2 (pp. 291-297)
- Non-linear features, or domain-specific similarity measurements are useful
- Dot products of non-linear features, or similarity measurements, can be written as kernel functions
  - Validity by positive semi-definiteness of kernel function
- Can have algorithm work in non-linear feature space without actually mapping inputs to feature space
  - Advantageous when feature space is high-dimensional