Kernel Methods

COMP9417, 22T2

Kernel Methods

Primal vs. Dual Algorithms

Primal vs. Dual Algorithms

The *dual* view of a problem is simply just another way to view a problem mathematically.

Primal vs. Dual Algorithms

The *dual* view of a problem is simply just another way to view a problem mathematically.

Instead of pure parameter based learning (i.e minimising a loss function etc.), dual algorithms introduce **instance-based** learning.

Primal vs. Dual Algorithms

The *dual* view of a problem is simply just another way to view a problem mathematically.

Instead of pure parameter based learning (i.e minimising a loss function etc.), dual algorithms introduce **instance-based** learning.

This is where we 'remember' mistakes in our data and adjust the corresponding weights accordingly.

We then use a *similarity function* or **kernel** in our predictions to weight the influence of the training data on the prediction.

In the primal problem, we typically learn parameters:

$$\mathbf{w} \in \mathbb{R}^p$$

meaning we learn parameters for each of the p features in our dataset.

In the primal problem, we typically learn parameters:

$$\mathbf{w} \in \mathbb{R}^p$$

meaning we learn parameters for each of the p features in our dataset.

In the dual problem, we typically learn parameters:

$$\alpha_i \qquad \text{for } i \in [1, n]$$

meaning we learn parameters for each of the n **data-points**.

In the primal problem, we typically learn parameters:

$$\mathbf{w} \in \mathbb{R}^p$$

meaning we learn parameters for each of the p features in our dataset.

In the dual problem, we typically learn parameters:

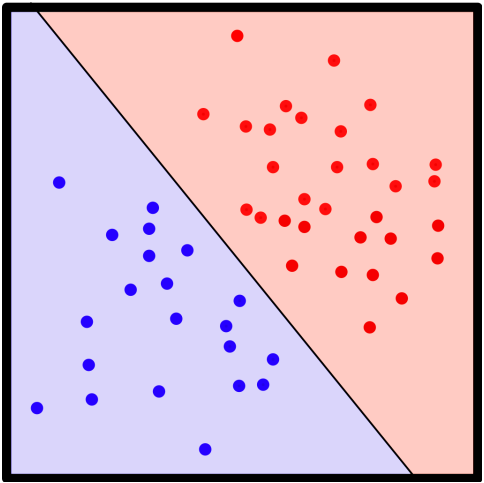
$$\alpha_i \qquad \text{for } i \in [1, n]$$

meaning we learn parameters for each of the n **data-points**.

α_i represents the *importance* of a data point (x_i, y_i).

What do we mean by importance?

What do we mean by importance?



The Dual/Kernel Perceptron

Recall the *primal* perceptron:

converged $\leftarrow 0$
while not *converged* **do**
    converged $\leftarrow 1$
    for $x_i \in X, y_i \in y$ **do**
        if $y_i w \cdot x_i \leq 0$ **then**
            $w \leftarrow w + \eta y_i x_i$
            converged $\leftarrow 0$
        end if
    end for
end while

The Dual/Kernel Perceptron

Recall the *primal* perceptron:

converged $\leftarrow 0$
while not *converged* **do**
  converged $\leftarrow 1$
  for $x_i \in X, y_i \in y$ **do**
    if $y_i w \cdot x_i \leq 0$ **then**
      $w \leftarrow w + \eta y_i x_i$
      converged $\leftarrow 0$
    end if
  end for
end while

If we define the number of iterations the perceptron makes as $K \in \mathbb{N}^+$ and assume $\eta = 1$. We can derive an expression for the final weight vector $w^{(K)}$:

The Dual/Kernel Perceptron

Recall the *primal* perceptron:

converged $\leftarrow 0$
while not *converged* **do**
  converged $\leftarrow 1$
  for $x_i \in X, y_i \in y$ **do**
    if $y_i w \cdot x_i \leq 0$ **then**
      $w \leftarrow w + \eta y_i x_i$
      converged $\leftarrow 0$
    end if
  end for
end while

If we define the number of iterations the perceptron makes as $K \in \mathbb{N}^+$ and assume $\eta = 1$. We can derive an expression for the final weight vector $w^{(K)}$:

$$w^{(K)} = \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbf{1}\{y_i w^{(j)} x_i \leq 0\} y_i x_i$$

We can simply our expression and take out the indicator variable:

$$
\begin{aligned}
w^{(K)} &= \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbf{1}\{y_i w^{(j)} x_i \leq 0\} y_i x_i \\
&= \sum_{i=1}^{N} \alpha_i y_i x_i
\end{aligned}
$$

where α_i is the number of times the perceptron makes a mistake on a data point (x_i, y_i).

If we sub in $w^{(K)} = \sum_{i=1}^{N} \alpha_i y_i x_i$. We get the algorithm for the **dual** perceptron.

If we sub in $w^{(K)} = \sum_{i=1}^{N} \alpha_i y_i x_i$. We get the algorithm for the **dual** perceptron.

converged $\leftarrow 0$
while not *converged* **do**
    converged $\leftarrow 1$
    for $x_i \in X, y_i \in y$ **do**
        if $y_i \sum_{j=1}^{N} \alpha_j y_j x_j \cdot x_i \leq 0$ **then**
            $\alpha_i \leftarrow \alpha_i + 1$
            converged $\leftarrow 0$
        end if
    end for
end while

Gram Matrix

The Gram matrix represents the *inner product* of two vectors.
For a dataset X we define $G = X^T X$. That is:

Gram Matrix

The Gram matrix represents the *inner product* of two vectors.
For a dataset X we define $G = X^T X$. That is:

$$G = \begin{bmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \cdots & \langle x_1, x_n \rangle \\ \langle x_2, x_1 \rangle & \langle x_2, x_2 \rangle & \cdots & \langle x_2, x_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_n, x_1 \rangle & \langle x_n, x_2 \rangle & \cdots & \langle x_n, x_n \rangle \end{bmatrix}$$

$$G_{i,j} = \langle x_i, x_j \rangle$$

Transformations

Transformations

How do we go about solving **non-linearly separable** datasets with linear classifiers?

Transformations

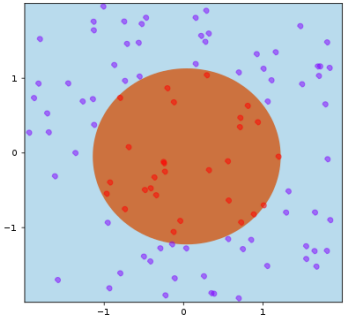
How do we go about solving **non-linearly separable** datasets with linear classifiers?

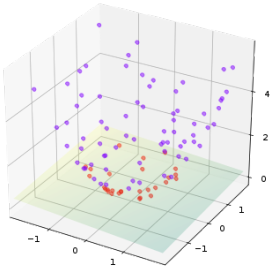
Project them to higher dimensional spaces through a transformation $\phi : \mathbb{R}^p \to \mathbb{R}^k$.

Transformations

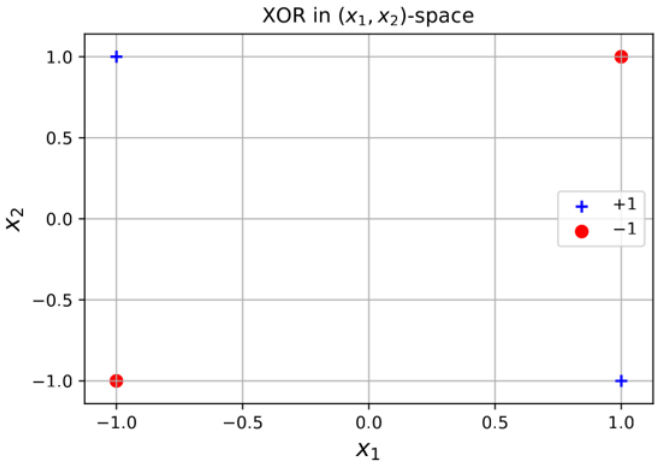
How do we go about solving **non-linearly separable** datasets with linear classifiers?

Project them to higher dimensional spaces through a transformation $\phi : \mathbb{R}^p \to \mathbb{R}^k$.





Let's revisit the XOR.



A solution:

A solution:

For our input vectors in the form $\mathbf{x} = [x_1, x_2]^T$, use a transformation:

$$\phi(\mathbf{x}) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}$$

For our dataset,

For our dataset,

$$\phi\left(\begin{bmatrix}1\\1\end{bmatrix}\right) = \begin{bmatrix}1\\\sqrt{2}\\\sqrt{2}\\1\\1\\\sqrt{2}\end{bmatrix} \quad \phi\left(\begin{bmatrix}-1\\-1\end{bmatrix}\right) = \begin{bmatrix}1\\-\sqrt{2}\\-\sqrt{2}\\1\\1\\\sqrt{2}\end{bmatrix} \quad \phi\left(\begin{bmatrix}-1\\1\end{bmatrix}\right) = \begin{bmatrix}1\\-\sqrt{2}\\\sqrt{2}\\1\\1\\-\sqrt{2}\end{bmatrix} \quad \phi\left(\begin{bmatrix}1\\-1\end{bmatrix}\right) = \begin{bmatrix}1\\\sqrt{2}\\-\sqrt{2}\\1\\1\\-\sqrt{2}\end{bmatrix}$$

For the negative class:

$$\phi\left(\begin{bmatrix}1\\1\end{bmatrix}\right)_{2,6} = \begin{bmatrix}\sqrt{2}\\\sqrt{2}\end{bmatrix}$$

$$\phi\left(\begin{bmatrix}-1\\-1\end{bmatrix}\right)_{2,6} = \begin{bmatrix}-\sqrt{2}\\\sqrt{2}\end{bmatrix}$$

For the positive class:

$$\phi\left(\begin{bmatrix}-1\\1\end{bmatrix}\right)_{2,6} = \begin{bmatrix}-\sqrt{2}\\-\sqrt{2}\end{bmatrix}$$

$$\phi\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)_{2,6} = \begin{bmatrix}\sqrt{2}\\-\sqrt{2}\end{bmatrix}$$

For the negative class:

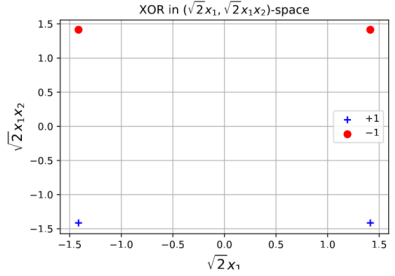
$$\phi\left(\begin{bmatrix}1\\1\end{bmatrix}\right)_{2,6} = \begin{bmatrix}\sqrt{2}\\\sqrt{2}\end{bmatrix}$$

$$\phi\left(\begin{bmatrix}-1\\-1\end{bmatrix}\right)_{2,6} = \begin{bmatrix}-\sqrt{2}\\\sqrt{2}\end{bmatrix}$$

For the positive class:

$$\phi\left(\begin{bmatrix}-1\\1\end{bmatrix}\right)_{2,6} = \begin{bmatrix}-\sqrt{2}\\-\sqrt{2}\end{bmatrix}$$

$$\phi\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)_{2,6} = \begin{bmatrix}\sqrt{2}\\-\sqrt{2}\end{bmatrix}$$



We may have a problem, recall the **dual perceptron**.

converged $\leftarrow 0$
while not *converged* **do**
  converged $\leftarrow 1$
  for $x_i \in X, y_i \in y$ **do**
    if $y_i \sum_{j=1}^{N} \alpha_j y_j x_j \cdot x_i \leq 0$ **then**
      $\alpha_i \leftarrow \alpha_i + 1$
      converged $\leftarrow 0$
    end if
  end for
end while

We may have a problem, recall the **dual perceptron**.

converged $\leftarrow 0$
while not *converged* **do**
  converged $\leftarrow 1$
  for $x_i \in X, y_i \in y$ **do**
    if $y_i \sum_{j=1}^{N} \alpha_j y_j \phi(x_j) \cdot \phi(x_i) \leq 0$ **then**
      $\alpha_i \leftarrow \alpha_i + 1$
      converged $\leftarrow 0$
    end if
  end for
end while

Recall the transformation $\phi : \mathbb{R}^p \to \mathbb{R}^k$.

Recall the transformation $\phi : \mathbb{R}^p \to \mathbb{R}^k$. For an arbitrarily large k,

$$G = \begin{bmatrix} \langle \phi(x_1), \phi(x_1) \rangle & \langle \phi(x_1), \phi(x_2) \rangle & \cdots & \langle \phi(x_1), x_n \rangle \\ \langle \phi(x_2), \phi(x_1) \rangle & \langle \phi(x_2), \phi(x_2) \rangle & \cdots & \langle \phi(x_2), \phi(x_n) \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \phi(x_n), \phi(x_1) \rangle & \langle \phi(x_n), \phi(x_2) \rangle & \cdots & \langle \phi(x_n), \phi(x_n) \rangle \end{bmatrix}$$

the Gram matrix becomes far too complex to compute.

The Kernel Trick

The Kernel Trick

An absolute mathemagical idea which allows us to calculate the values of the Gram matrix for cheap.

Recall the transformation to the XOR data:

$$\phi(\mathbf{x}) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}$$

The Kernel Trick

An absolute mathemagical idea which allows us to calculate the values of the Gram matrix for cheap.

Recall the transformation to the XOR data:

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix} \begin{bmatrix} 1 \\ \sqrt{2}y_1 \\ \sqrt{2}y_2 \\ y_1^2 \\ y_2^2 \\ \sqrt{2}y_1y_2 \end{bmatrix}$$

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2x_1y_1 + 2x_2y_2 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2$$

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2x_1y_1 + 2x_2y_2 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2$$
$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2(x_1y_1 + x_2y_2) + (x_1y_1 + x_2y_2)^2$$

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2x_1y_1 + 2x_2y_2 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2$$
$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2(x_1y_1 + x_2y_2) + (x_1y_1 + x_2y_2)^2$$
$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2$$

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2x_1y_1 + 2x_2y_2 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2$$
$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2(x_1y_1 + x_2y_2) + (x_1y_1 + x_2y_2)^2$$
$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2$$

Say we define a *kernel*: $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2$

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2x_1y_1 + 2x_2y_2 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2$$
$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2(x_1y_1 + x_2y_2) + (x_1y_1 + x_2y_2)^2$$
$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2$$

Say we define a *kernel*: $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2$

So our Gram matrix is:

$$G = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{bmatrix}$$

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2x_1y_1 + 2x_2y_2 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2$$
$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = 1 + 2(x_1y_1 + x_2y_2) + (x_1y_1 + x_2y_2)^2$$
$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2$$

Say we define a *kernel*: $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2$

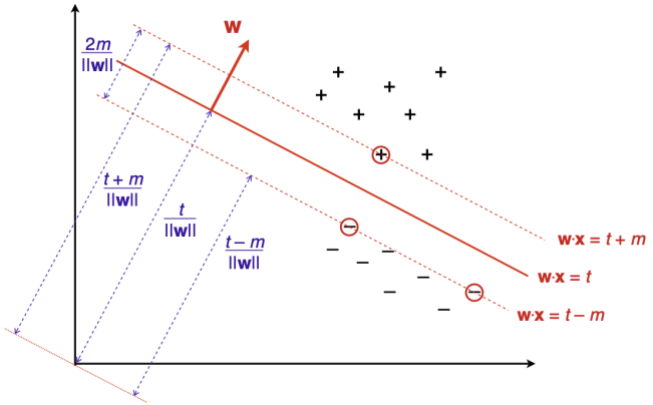
So our Gram matrix is:

$$G = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{bmatrix}$$

Why is this useful?

Support Vector Machines

Support Vector Machines



The basic SVM is a linear classifier defined by:

$$\arg\min_{w,t} \frac{1}{2}\|w\|^2 \qquad \text{subject to } y_i(\langle x_i, w\rangle - t) \geq m$$

where t is the line's intercept, and we a consider a margin m. Typically, we'll see $m = 1$ for a standardised dataset.

The basic SVM is a linear classifier defined by:

$$\arg\min_{w,t} \frac{1}{2}\|w\|^2 \qquad \text{subject to } y_i(\langle x_i, w\rangle - t) \geq m$$

where t is the line's intercept, and we a consider a margin m. Typically, we'll see $m = 1$ for a standardised dataset.

This formulation means that we find the **maximal margin** classifier for the dataset.

Aside: Lagrangian Dual Problem

Say we have a problem as follows:

$$\max_{x,y} xy \qquad \text{subject to } x + y = 4$$

we can also consider the constraint as $x + y - 4 = 0$.

Aside: Lagrangian Dual Problem

Say we have a problem as follows:

$$\max_{x,y} xy \qquad \text{subject to } x + y = 4$$

we can also consider the constraint as $x + y - 4 = 0$.

We can set up the Lagrangian dual and *move* the constraint into the function itself:

$$\Lambda(x, y, \lambda) = xy + \lambda(x + y - 4)$$

Aside: Lagrangian Dual Problem

Say we have a problem as follows:

$$\max_{x,y} xy \qquad \text{subject to } x + y = 4$$

we can also consider the constraint as $x + y - 4 = 0$.

We can set up the Lagrangian dual and *move* the constraint into the function itself:

$$\Lambda(x, y, \lambda) = xy + \lambda(x + y - 4)$$

To solve this, we can calculate $\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial y}$ and $\frac{\partial L}{\partial \lambda}$ and solve the remaining system of equations.

The General Form of a Dual Problem

If we have a problem:

$$\arg\min_x f(x)$$
$$\text{subject to } g_i(x) \leq 0, \qquad i \in \{1, \ldots, n\}$$

The General Form of a Dual Problem

If we have a problem:

$$\arg\min_{x} f(x)$$
$$\text{subject to } g_i(x) \leq 0, \qquad i \in \{1, \ldots, n\}$$

The general *dual* problem is:

$$\Lambda(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^{n} \lambda_i g_i(x_i)$$

The Dual Problem for SVM

If we take the general SVM problem ($m = 1$):

$$\arg\min_{w,t} \frac{1}{2}\|w\|^2 \qquad \text{subject to } y_i(\langle x_i, w\rangle - t) \geq 1$$

The Dual Problem for SVM

If we take the general SVM problem ($m = 1$):

$$\arg\min_{w,t} \frac{1}{2}\|w\|^2 \qquad \text{subject to } y_i(\langle x_i, w\rangle - t) - 1 \geq 0$$

The Dual Problem for SVM

If we take the general SVM problem ($m = 1$):

$$\arg\min_{w,t} \frac{1}{2}\|w\|^2 \qquad \text{subject to } y_i(\langle x_i, w\rangle - t) - 1 \geq 0$$

From the general form, we can take the vector α to form the dual problem:

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 + \left(-\sum_{i=1}^{n} \alpha_i y_i(\langle x_i, w\rangle - t) - 1)\right)$$

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 + \left(- \sum_{i=1}^{n} \alpha_i y_i (\langle x_i, w \rangle - t) - 1) \right)$$

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 + \left(- \sum_{i=1}^{n} \alpha_i y_i (\langle x_i, w \rangle - t) - 1) \right)$$

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i (w \cdot x_i) + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 + \left(- \sum_{i=1}^{n} \alpha_i y_i (\langle x_i, w \rangle - t) - 1) \right)$$

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i (w \cdot x_i) + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

Let's try and optimise the Lagrangian Λ w.r.t w,

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

Let's try and optimise the Lagrangian Λ w.r.t w,

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\frac{\partial \Lambda}{\partial w} = \frac{1}{2} 2w - \sum_{i=1}^{n} \alpha_i y_i x_i$$

Let's try and optimise the Lagrangian Λ w.r.t w,

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\begin{aligned} \frac{\partial \Lambda}{\partial w} &= \frac{1}{2} 2w - \sum_{i=1}^{n} \alpha_i y_i x_i \\ &= w - \sum_{i=1}^{n} \alpha_i y_i x_i \end{aligned}$$

Let's try and optimise the Lagrangian Λ w.r.t w,

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\begin{aligned} \frac{\partial \Lambda}{\partial w} &= \frac{1}{2} 2w - \sum_{i=1}^{n} \alpha_i y_i x_i \\ &= w - \sum_{i=1}^{n} \alpha_i y_i x_i \end{aligned}$$

We can see that at $\frac{\partial \Lambda}{\partial w} = 0$

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

Repeating a similar process for t,

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

Repeating a similar process for t,

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\frac{\partial \Lambda}{\partial t} = \sum_{i=1}^{n} \alpha_i y_i$$

Repeating a similar process for t,

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\frac{\partial \Lambda}{\partial t} = \sum_{i=1}^{n} \alpha_i y_i$$

We can see that at $\frac{\partial \Lambda}{\partial t} = 0$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

The Dual Problem for SVM

We've derived that for an optimal solution, $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $w = \sum_{i=1}^{n} \alpha_i y_i x_i$

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

The Dual Problem for SVM

We've derived that for an optimal solution, $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $w = \sum_{i=1}^{n} \alpha_i y_i x_i$

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\Lambda(w, \alpha) = \frac{1}{2} w^T w - w^T w + \sum_{i=1}^{n} \alpha_i$$

The Dual Problem for SVM

We've derived that for an optimal solution, $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $w = \sum_{i=1}^{n} \alpha_i y_i x_i$

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\Lambda(w, \alpha) = \frac{1}{2} w^T w - w^T w + \sum_{i=1}^{n} \alpha_i$$

$$\Lambda(w, \alpha) = -\frac{1}{2} w^T w + \sum_{i=1}^{n} \alpha_i$$

The Dual Problem for SVM

We've derived that for an optimal solution, $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $w = \sum_{i=1}^{n} \alpha_i y_i x_i$

$$\Lambda(w, t, \alpha) = \frac{1}{2}\|w\|^2 - w \cdot \sum_{i=1}^{n} \alpha_i y_i x_i + t \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\Lambda(w, \alpha) = \frac{1}{2} w^T w - w^T w + \sum_{i=1}^{n} \alpha_i$$

$$\Lambda(w, \alpha) = -\frac{1}{2} w^T w + \sum_{i=1}^{n} \alpha_i$$

$$\Lambda(\alpha) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^{n} \alpha_i$$

Our final problem now has relaxed constraints:

$$\Lambda(\alpha) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^{n}\alpha_i$$

$$\text{subject to } \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$\alpha_i \geq 0 \text{ for } i = 1, \ldots, n$$

Soft Margin SVM

Our current model is a **maximum** or hard margin classifier. To allow for errors *within* the supporting hyperplanes, we can redefine the primal problem as:

$$\arg\min_{w,t,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \qquad \text{subject to } y_i(\langle x_i, w\rangle - t) \geq 1 - \xi_i \text{ where } \xi_i \geq 0$$

we typically take ξ_i as the *hinge loss* of a point.

A Slight Extension: Hinge Loss

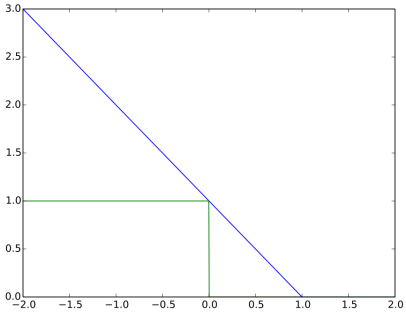
We define hinge loss for a data point at (x_i, y_i) as:

$$\xi_i = \max(0, 1 - y_i(w^T x_i - b))$$

A Slight Extension: Hinge Loss

We define hinge loss for a data point at (x_i, y_i) as:

$$\xi_i = \max(0, 1 - y_i(w^T x_i - b))$$



So, the function we minimise is essentially:

$$\arg\min_{w,t,\xi} \frac{1}{2}\|w\|^2 + C \max(0, 1 - y_i(w^T x_i - b))$$

$$\text{subject to } y_i(\langle x_i, w\rangle - t) \geq 1 - \max(0, 1 - y_i(w^T x_i - b))$$

Extension: The RBF Kernel

Extension: The RBF Kernel

A popular Kernel is the Radial Basis Function kernel, defined below:

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

for scalar values:

$$K(x, y) = \exp\left(-\frac{(x - y)^2}{2\sigma^2}\right)$$

$$K(x, y) = \exp\left(\frac{(x - y)^2}{2\sigma^2}\right)$$

$$K(x, y) = \exp\left(\frac{(x - y)^2}{2\sigma^2}\right)$$
$$= \exp\left(\frac{-x^2 + 2xy - y^2}{2\sigma^2}\right)$$

$$
\begin{aligned}
K(x, y) &= \exp\left(\frac{(x-y)^2}{2\sigma^2}\right) \\
&= \exp\left(\frac{-x^2 + 2xy - y^2}{2\sigma^2}\right) \\
&= \exp\left(\frac{-x^2}{2\sigma^2}\right) \exp\left(\frac{-y^2}{2\sigma^2}\right) \exp\left(\frac{xy}{\sigma^2}\right)
\end{aligned}
$$

$$
\begin{aligned}
K(x,y) &= \exp\left(\frac{(x-y)^2}{2\sigma^2}\right) \\
&= \exp\left(\frac{-x^2+2xy-y^2}{2\sigma^2}\right) \\
&= \exp\left(\frac{-x^2}{2\sigma^2}\right)\exp\left(\frac{-y^2}{2\sigma^2}\right)\exp\left(\frac{xy}{\sigma^2}\right) \\
&= \exp\left(\frac{-x^2}{2\sigma^2}\right)\exp\left(\frac{-y^2}{2\sigma^2}\right)\sum_{i=1}^{\infty}\frac{(xy)^k}{\sigma^{2k}k!}
\end{aligned}
$$

By definition

$$\langle \phi(x), \phi(y) \rangle = \exp\left(\frac{-x^2}{2\sigma^2}\right) \exp\left(\frac{-y^2}{2\sigma^2}\right) \sum_{i=1}^{\infty} \frac{(xy)^k}{\sigma^{2k} k!}$$

So, our basis transformation is:

$$\phi(x) = \exp\left(\frac{-x^2}{2\sigma^2}\right) \sum_{i=1}^{\infty} \frac{x^k}{\sigma^k \sqrt{k!}}$$

What does this represent?

By definition

$$\langle \phi(x), \phi(y) \rangle = \exp\left(\frac{-x^2}{2\sigma^2}\right) \exp\left(\frac{-y^2}{2\sigma^2}\right) \sum_{i=1}^{\infty} \frac{(xy)^k}{\sigma^{2k} k!}$$

So, our basis transformation is:

$$\phi(x) = \exp\left(\frac{-x^2}{2\sigma^2}\right) \sum_{i=1}^{\infty} \frac{x^k}{\sigma^k \sqrt{k!}}$$

What does this represent? A projection to infinite dimensions!