Bankruptcy Example

1-Nearest Neighbor Hypothesis

Decision Tree Hypothesis

Linear Hypothesis
Linearly Separable

Not Linearly Separable

Not Linearly Separable

Not Linearly Separable
Linear Hypothesis Class
• Equation of a hyperplane in the feature space
  \[ \mathbf{w} \cdot \mathbf{x} + b = 0 \]
  \[ \sum_{j=1}^{n} w_j x_j + b = 0 \]
• \( \mathbf{w} \), \( b \) are to be learned

Hyperplane: Geometry
• Equation of a hyperplane in the feature space
  \[ \mathbf{w} \cdot \mathbf{x} + b = 0 \]
  \[ \sum_{j=1}^{n} w_j x_j + b = 0 \]
• \( \mathbf{w} \), \( b \) are to be learned

A useful trick: let \( x_0 = 1 \) and \( w_0 = b \)
\[ \overrightarrow{\mathbf{w}} \cdot \overrightarrow{x} = 0 \]
\[ \sum_{j=0}^{n} w_j x_j = 0 \]
**Hyperplane: Geometry**

\[ \mathbf{w} \cdot \mathbf{x} + b \]

signed perpendicular distance of point \( \mathbf{x} \) to hyperplane.

recall: \( \mathbf{a} \cdot \mathbf{b} = \| \mathbf{a} \| \| \mathbf{b} \| \cos \theta \)

**Linear Classifier**

\[ h(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) = \text{sign}(\mathbf{w} \cdot \mathbf{x}) \]

outputs +1 or -1

Margin:

\[ y_i = y'(\mathbf{w} \cdot \mathbf{x}^i + b) = y'\mathbf{w} \cdot \mathbf{x}^i \]

proportional to perpendicular distance of point \( \mathbf{x}^i \) to hyperplane.

\[ y_i > 0 : \text{point is correctly classified (sign of distance = } y') \]

\[ y_i < 0 : \text{point is incorrectly classified (sign of distance = } y') \]
Perceptron Algorithm
Rosenblatt, 1956

• Pick initial weight vector (including b), e.g. [0 ... 0]
• Repeat until all points correctly classified
  • Repeat for each point
    – Calculate margin ( ) for point i
    – If margin > 0, point is correctly classified
    – Else change weights to increase margin; change in weight
      proportional to

• Note that, if \( y = 1 \)
  if \( x_j > 0 \) then \( w_j \) increased (increases margin)
  if \( x_j < 0 \) then \( w_j \) decreased (increases margin)
• And, similarly for \( y = -1 \)
• Guaranteed to find separating hyperplane if one exists
• Otherwise, data are not linearly separable, loops forever

Perceptron Algorithm
Rosenblatt, 1956

• Pick initial weight vector (including b), e.g. [0 ... 0]
• Repeat until all points correctly classified
  • Repeat for each point
    – Calculate margin ( ) for point i
    – If margin > 0, point is correctly classified
    – Else change weights to increase margin; change in weight
      proportional to

• Note that, if \( y = 1 \)
  if \( x_j > 0 \) then \( w_j \) increased (increases margin)
  if \( x_j < 0 \) then \( w_j \) decreased (increases margin)
• And, similarly for \( y = -1 \)

Final Answer:
\[ w = [-2.2, 0.94, 0.4] \]

Initial Guess:
\[ w = [0, 0, 0, 0] \]
Gradient Ascent

• Why pick $y'x$ as increment to weights?
• To maximize scalar function of one variable $f(w)$
  • Pick initial $w$
  • Change $w$ to $w + \eta \frac{df}{dw}$ ($\eta > 0$, small)
  • until $f$ stops changing ($\frac{df}{dw} = 0$)

Gradient Ascent/Descent

• To maximize $f(w)$ $\nabla_w f = \left[ \frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_n} \right]$
• Pick initial $w$
• Change $w$ to $w + \eta \nabla_w f$ ($\eta > 0$, small)
• until $f$ stops changing ($\nabla_w f = 0$)
• Finds local maximum; global maximum if function is globally convex.

Perceptron Training via Gradient Descent

• Maximize sum of margins of misclassified points
  $f(w) = \sum_{i \text{ misclassified}} y_i w^T x_i$
  $\nabla_w f = \sum_{i \text{ misclassified}} y_i x_i$
Perceptron Training via Gradient Descent

- Maximize sum of margins of misclassified points
  \[ f(w) = \sum_{i \text{ misclassified}} y^i w \cdot x^i \]
  \[ \nabla f = \sum_{i \text{ misclassified}} y^i x^i \]

- Off-line training: Compute gradient as sum over all training points.
- On-line training: Approximate gradient by one of the terms in the sum: \( y^i x^i \)

### Perceptron Algorithm Bankruptcy Data

**Rate** \( \eta = 0.1 \)

<table>
<thead>
<tr>
<th>Class</th>
<th>Feature 1</th>
<th>Feature 2</th>
<th>Feature 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>-1.4</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>-1.6</td>
<td>0.6</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td>-1.6</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>5</td>
<td>-1.7</td>
<td>0.6</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**Initial Guess:** \( w = [-1.0 1.0 1.0] \)

**Final Answer:** \( w = [-1.7 0.8 0.3] \)

### Dual Form

Assume initial weights are 0; \( \eta > 0 \)

- \( a_i = \text{count of mistakes on point } i \) during training

\[
W = \frac{\eta}{m} \sum a_i y^i x^i
\]
Dual Form

Assume initial weights are 0; rate=$\eta>0$

\[
\begin{align*}
5 \cdot (1.0 0.2 6.0) \\
3 \cdot (1.0 1.1 3.0) \\
-9 \cdot (1.0 0.2 3.0) \\
-1 \cdot (1.0 0.7 2.0) \\
-4 \cdot (1.0 0.5 4.0) \\
-1 \cdot (1.0 1.7 1.0) \\
\end{align*}
\]

\[-7.0 \cdot (-1.9 -7.0) \cdot 0.1 = -0.7 -0.19 -0.7\]

\[
h(x) = \text{sign}(w \cdot x) = \text{sign}(\sum_{i=1}^{m} \alpha_i y_i x_i) \]

Perceptron Training

• $\alpha = 0$

• Repeat until all points correctly classified
  • Repeat for each point $i$
    - Calculate margin $\sum_{j=1}^{m} \alpha_i y_i x_i$
    - If margin $> 0$, point is correctly classified
    - Else increment $\alpha_i$
  • Return $w = \sum_{j=1}^{m} \alpha_i y_i x_i$

• If data is not linearly separable, the $\alpha_i$ grow without bound

Which Separator?

Maximize the margin to closest points
Which Separator?
Maximize the margin to closest points

Margin of a point
\[ \gamma' = y'(w \cdot x' + b) \]
• proportional to perpendicular distance of point \( x' \) to hyperplane

Margin of a point
\[ \gamma' = y'(w \cdot x' + b) \]
• proportional to perpendicular distance of point \( x' \) to hyperplane

Margin
\[ \gamma' = y'(w \cdot x' + b) \]
• Scaling \( w \) changes value of margin but not actual distances to separator (geometric margin)
• Pick the margin to closest positive and negative points to be 1
  + \[ 1(w \cdot x' + b) = 1 \]
  - \[ 1(w \cdot x' + b) = -1 \]
Margin

• Pick the margin to closest positive and negative points to be 1
  \[ +\mathbf{1}(\mathbf{w} \cdot \mathbf{x} + b) = 1 \]
  \[ -\mathbf{1}(\mathbf{w} \cdot \mathbf{x} + b) = 1 \]
• Combining these
  \[ \mathbf{w} \cdot (\mathbf{x}^+ - \mathbf{x}^-) = 2 \]
• Dividing by length of \( \mathbf{w} \) gives perpendicular distance between dashed lines (2x geometric margin)
  \[ \frac{\mathbf{w}}{||\mathbf{w}||} (\mathbf{x}^+ - \mathbf{x}^-) = \frac{2}{||\mathbf{w}||} \]

Picking \( \mathbf{w} \) to Maximize Margin

• Pick \( \mathbf{w} \) to maximize geometric margin
  \[ \frac{2}{||\mathbf{w}||} \]
• or, equivalently, minimize
  \[ ||\mathbf{w}|| = \sqrt{\mathbf{w} \cdot \mathbf{w}} \]
• or, equivalently, minimize
  \[ \frac{1}{2} ||\mathbf{w}||^2 - \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \frac{1}{2} \sum w_i^2 \]

Picking \( \mathbf{w} \) to Maximize Margin

• Pick \( \mathbf{w} \) to maximize geometric margin
  \[ \frac{2}{||\mathbf{w}||} \]
• or, equivalently, minimize
  \[ ||\mathbf{w}|| = \sqrt{\mathbf{w} \cdot \mathbf{w}} \]
• or, equivalently, minimize
  \[ \frac{1}{2} ||\mathbf{w}||^2 - \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \frac{1}{2} \sum w_i^2 \]

Constrained Optimization

\[ \min_{\frac{1}{2} ||\mathbf{w}||} \text{subject to } y_i'(\mathbf{w} \cdot \mathbf{x}^i + b) - 1 \geq 0, \ \forall_i \]
A Simple Example

\[
\min_\alpha \frac{1}{2} w^2 \quad \text{subject to } w x - 1 \geq 0
\]

Incorporate constraint

\[
\min L(w) = \min_\alpha \left( \frac{1}{2} w^2 - \alpha(w x - 1) \right)
\]

As \( \alpha \) increases, minimal \( w \) gets farther from original optimum

\( \alpha = 1 \) satisfies constraint with minimal distortion

Lagrangian

\[
\min L(w) = \min_\alpha \left( \frac{1}{2} w^2 - \alpha(w x - 1) \right)
\]

Find minimum of \( L \) wrt \( w \)

\[
\frac{\partial L(w)}{\partial w} = w - \alpha x = 0
\]

\( w^* = \alpha x \)
Lagrangian

\[
\min_w L(w) = \min_w \left( \frac{1}{2} w^2 - \alpha(wx - 1) \right)
\]

Find minimum of \( L \) wrt \( w \)

\[
\frac{dL(w)}{dw} = w - \alpha x = 0
\]

\( w = \alpha x \)

Substituting back in \( L \) gives us Lagrangian dual

\[
L(\alpha) = \frac{1}{2} (\alpha x)^2 - \alpha((\alpha x) - 1) = \frac{1}{2} \alpha^2 x^2 - \alpha x - \alpha
\]

\[ L(\alpha) = \alpha + \frac{1}{2} \alpha^2 x^2 \]

Lagrangian Dual

\[
L(\alpha) = \alpha - \frac{1}{2} \alpha^2 x^2
\]

Find maximum of \( L \) wrt \( \alpha \)

\[
\frac{dL(\alpha)}{d\alpha} = 1 - \alpha x^2 - 1 - w^* x = 0
\]

Recall original constraint is \( wx - 1 \geq 0 \)

Constraint is satisfied with equality at maximum of \( L(\alpha) \)

In general, since \( \alpha \geq 0 \), either

- \( \alpha = 0 \) constraint is satisfied with no distortion at optimum \( w \)
- \( \alpha > 0 \) constraint is satisfied with equality
**A Simple Example**

Summary

Original Problem: \[ \min_\mathbf{w} \frac{1}{2} \mathbf{w}^T \mathbf{w} \text{ subject to } \mathbf{w}^T \mathbf{x} - 1 \geq 0 \]

Introduce Lagrange multipliers: \[ \min_{\mathbf{w}} L(\mathbf{w}) = \min_{\mathbf{w}} \left( \frac{1}{2} \mathbf{w}^T \mathbf{w} - \alpha (\mathbf{w}^T \mathbf{x} - 1) \right) \]

Find minimum of \( L(\mathbf{w}) \) wrt \( \mathbf{w} \)

\[ \mathbf{w}^* = \alpha \mathbf{x} \]

Substituting back in \( L \) gives us Lagrangian dual

\[ L(\alpha) = \alpha - \frac{1}{2} \alpha^2 x^2 \]

Find maximum of \( L(\alpha) \) wrt \( \alpha \)

\[ \frac{dL(\alpha)}{d\alpha} \cdot 1 - \alpha \alpha^2 = 0 \]

Use optimal \( \alpha \) to get optimal \( \mathbf{w} \)

---

**Constrained Optimization**

\[ \min_{\mathbf{w}} \frac{1}{2} \| \mathbf{w} \|^2 \text{ subject to } \mathbf{y}^T (\mathbf{w} \cdot \mathbf{x}^T + b) - 1 \geq 0, \quad \forall_i \]

Convert to unconstrained optimization by incorporating the constraints as an additional term

\[ \min_{\mathbf{w}} \left( \frac{1}{2} \| \mathbf{w} \|^2 - \sum_i \alpha_i \left[ \mathbf{y}^T (\mathbf{w} \cdot \mathbf{x}^T + b) - 1 \right] \right) \quad \alpha_i \geq 0, \quad \forall_i \]

To minimize expression:

- Minimize first (original) term, and maximize second (constraint) term
- Since \( \alpha_i > 0 \), encourages constraints to be satisfied but we want least “distortion” of original term...
**Constrained Optimization**

\[
\min_w \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad y'(w \cdot x^i + b) - 1 = 0, \ \forall i,
\]

Convert to unconstrained optimization by incorporating the constraints as an additional term:

\[
\min \left\{ \frac{1}{2} \|w\|^2 - \sum \alpha_i [y'(w \cdot x^i + b) - 1] \right\} \quad \alpha_i \geq 0, \ \forall i
\]

Lagrange multipliers

To minimize expression:
- minimize first (original) term, and
- maximize second (constraint) term

since \(\alpha_i > 0\), encourages constraints to be satisfied but we want least “distortion” of original term...

**Method of Lagrange multipliers**

**Maximizing the Margin**

\[
L(w, b) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y'(w \cdot x^i + b) - 1]
\]

Minimized when:

\[
w^* = \sum \alpha_i y^i x^i \quad \sum \alpha_i y^i = 0
\]

Substituting \(w^*\) into \(L\) yields dual Lagrangian:

\[
L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^i y^j x^i \cdot x^j
\]

Only dot products of the feature vectors appear.
Maximizing Margin in General
\[ L(w, b) = \frac{1}{2} \|w\|^2 - \sum_i y_i (w \cdot x_i + b) - 1 \]

\[ \frac{\partial L(w, b)}{\partial w} \]
\[ w = \sum_i y_i x_i' \]
Recall:
\[ \frac{\partial f}{\partial w} = \left[ \frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_n} \right] \]

\[ \|w\|^2 - w \cdot w = \sum_i w_i^2 \implies \frac{\partial \|w\|^2}{\partial w} = 2w \]
\[ w \cdot x' = \sum_i w_i x'_i \implies \frac{\partial w \cdot x'}{\partial w} = x' \]

Maximizing Margin in General
\[ L(w, b) = \frac{1}{2} \|w\|^2 - \sum_i y_i (w \cdot x_i + b) - 1 \]

\[ \frac{\partial L(w, b)}{\partial w} = 0 \]
\[ \frac{\partial L(w, b)}{\partial b} = 0 \]
Recall:
\[ \frac{\partial f}{\partial w} = \left[ \frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_n} \right] \]
Maximizing Margin in General

\[ L(w, b) = \frac{1}{2} |w|^2 - \sum_i y_i (w \cdot x_i + b) - 1 \]

Substituting: \[ w = \sum_i \alpha_i y_i x_i \]

Using: \[ \sum_i \alpha_i y_i = 0 \]

Dual Lagrangian

\[ \max \sum_i \alpha_i \text{ subject to } \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0, \forall i \]

In general, since \( \alpha \geq 0 \), either
- \( \alpha = 0 \): constraint is satisfied with no distortion at optimum \( w \)
- \( \alpha > 0 \): constraint is satisfied with equality (in this case \( x \) is known as a support vector)

\[ w = \sum_i \alpha_i y_i x_i \]

\[ b = \frac{1}{y_i - w \cdot x_i} \]
**Dual Lagrangian**

\[ \max L(\alpha) \text{ subject to } \sum \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0, \forall i \]

In general, since \( \alpha_i \geq 0 \), either
- \( \alpha_i = 0 \): constraint is satisfied with no distortion at optimum \( w \)
- \( \alpha_i > 0 \): constraint is satisfied with equality (\( x_i \) is known as a support vector)

- \( w^* = \sum \alpha_i y_i x_i \)
- Can be found using quadratic programming or gradient ascent

**SVM Classifier**

- Given unknown vector \( u \), predict class (1 or -1) as follows:
  \[
  h(u) = \text{sign} \left( \sum \alpha_i y_i x_i \cdot u + b \right)
  \]
- The sum is over \( k \) support vectors

**Bankruptcy Example**

\( \alpha_i y_i \) for support vectors are non-zero, all others are zero.

**Key Points**

- Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.
Key Points

• Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.
• Exclusive reliance on dot products enables approach to non-linearly-separable problems.

• The classifier depends only on the support vectors, not on all the training points.
• Max margin lowers hypothesis variance.

• The optimal classifier is defined uniquely – there are no "local maxima" in the search space
• Polynomial in number of data points and dimensionality
Not Linearly Separable?

- Require $0 \leq \alpha_i \leq C$
- $C$ specified by user; controls tradeoff between size of margin and classification errors
- $C = 1$ for separable case

C Change

Example: Linearly Separable

Another example: Not linearly separable
Isn’t a linear classifier very limiting?

Not separable?
Try a higher dimensional space!

Important: Linear separator in transformed feature space maps into non-linear separator in original feature space.

Not separable with 2D line
Separable with 3D plane

What you need

• To get into the new feature space, you use \( \phi(x') \)
• The transformation can be to a higher-dimensional feature space and may be non-linear in the feature values.

What you need

• To get into the new feature space, you use \( \phi(x') \)
• The transformation can be to a higher-dimensional feature space and may be non-linear in the feature values.
• Recall that SVM’s only use dot products of the data, so
• To optimize classifier, you need \( \phi(x') \cdot \phi(x') \)
• To run classifier, you need \( \phi(x) \cdot \phi(w) \)
• So, all you need is a way to compute dot products in transformed space as a function of vectors in original space!
The "Kernel Trick"

- If dot products can be efficiently computed by
  \[ \phi(x') \cdot \phi(x') = K(x', x') \]
- Then, all you need is a function on low-dim inputs
  \[ K(x', x') \]
- You don't need ever to construct high-dimensional
  \[ \phi(x') \]

Standard Choices For Kernels

- No change (linear kernel)
  \[ \phi(x') \cdot \phi(x') = K(x', x') = x' \cdot x' \]

Standard Choices For Kernels

- No change (linear kernel)
  \[ \phi(x') \cdot \phi(x') = K(x', x') = x' \cdot x' \]
- Polynomial kernel (n\(^{th}\) order)
  \[ K(x', x') = (1 + x' \cdot x')^n \]

Polynomial Kernel Example

(one feature)

Not separable
Polynomial Kernel Example

(one feature)

Polynomial Kernel

- Polynomial kernel for \( n = 2 \) and features \( \mathbf{x} = [x_1, x_2] \)
  
  \[ K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x} \cdot \mathbf{z})^2 \]

  is equivalent to the following feature mapping:

  \[ \phi(x) = [x_1^2, x_2^2, x_1, x_2] \]

  - We can verify that:
    
    \[ \phi(x) \cdot \phi(z) = x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1 \]
    
    \[ = (1 + x_1z_1 + x_2z_2)^2 \]
    
    \[ = (1 + \mathbf{x} \cdot \mathbf{z})^2 \]
    
    \[ = K(\mathbf{x}, \mathbf{z}) \]

Standard Choices For Kernels

- No change (linear kernel)
  \[ \phi(x') \cdot \phi(x') = K(x', x') = x' \cdot x' \]

- Polynomial kernel (\( n \)th order)
  \[ K(x', x') = (1 + x' \cdot x')^n \]

- Radial basis kernel (\( \sigma \) is standard deviation)
  \[ K(x', x') = e^{-\frac{(x' - x')^2}{2\sigma^2}} = e^{-\frac{(x' - x')^2}{2\sigma^2}} \]
Radial-basis kernel

- Classifier based on sum of Gaussian bumps with standard deviation $\sigma$, centered on support vectors.

$$ h(u) = \text{sign}[h'(u)] $$

$$ h'(u) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, u) + b $$

$$ K(x', u) = e^{-\frac{(x' - u)^2}{2\sigma^2}} $$
Radial-basis kernel (large $\sigma$)

Images by Patrick Winston

Another radial-basis example (small $\sigma$)

Images by Patrick Winston

Cross-Validation Error

- Does mapping to a very high-dimensional space lead to over-fitting?
- Generally, no, thanks to the fact that only the support vectors determine the decision surface.

Cross-Validation Error

- Does mapping to a very high-dimensional space lead to over-fitting?
- Generally, no, thanks to the fact that only the support vectors determine the decision surface.
- The expected leave-one-out cross-validation error depends on number of support vectors, not dimensionality of feature space.

\[
\text{Expected CV error} = \frac{\text{Expected # support vectors}}{\text{# training samples}}
\]

- If most data points are support vectors, a sign of possible overfitting, independent of the dimensionality of feature space.
Summary
• A single global optimum
  • Quadratic programming or gradient descent
• Fewer parameters
  • C and kernel parameters (n for polynomial, $\sigma$ for radial basis kernel)
• Kernel
  • Quadratic minimization depends only on dot products of sample vectors
  • Recognition depends only on dot products of unknown vector with sample vectors
  • Reliance on only dot products enables efficient feature mapping to higher-dimensional spaces where linear separation is more effective.

Real Data
• Wisconsin Breast Cancer Data
  • 9 features
  • $C=1$
  • 37 support vectors are used from 512 training data points
  • 12 prediction errors on training set (98% accuracy)
  • 96% accuracy on 171 held out points
  • Essentially same performance as nearest neighbors and decision trees
  • Don’t expect such good performance on every data set.
**Success Stories**

- Gene microarray data
  - outperformed all other classifiers
  - specially designed kernel
- Text categorization
  - linear kernel in >10,000 D input space
  - best prediction performance
  - 35 times faster to train than next best classifier (decision trees)
- Many others:

**Training Max Margin Classifier**

\[
\max_{\alpha} L(\alpha) \quad \text{subject to} \quad \sum y_i \alpha_i = 0 \quad \text{and} \quad \alpha_i \geq 0, \forall i
\]

In general, since \(\alpha_i \geq 0\), either
- \(\alpha_i = 0\) constraint is satisfied with no distortion at optimum \(w\) or
- \(\alpha_i > 0\) constraint is satisfied with equality (in this case \(x^i\) is known as a support vector)

Values of \(\alpha_i\) at maximum of \(L(\alpha)\) are used to find weights \(w\)

\[
w^* = \sum \alpha_i y^i x^i
\]

Remember dual perceptron!
Training Max Margin Classifier

$$\max_{\alpha} L(\alpha) \text{ subject to } \sum \alpha y = 0 \text{ and } \alpha \geq 0, \forall i$$

In general, since $$\alpha_i = 0$$, either
- $$\alpha_i = 0$$ constraint is satisfied with no distortion at optimum w
- $$\alpha_i > 0$$ constraint is satisfied with equality (in this case x is known as a support vector)

Values of $$\alpha_i$$ at maximum of $$L(\alpha)$$ are used to find weights w

$$w^* = \sum \alpha_i y x^i$$  
Remember dual perceptron!

For support vectors, constraints hold at equality, so we can solve for b (average for all support vectors)

$$y^i (w^* x^i + b) - 1 = 0$$

To the Max

$$\max_{\hat{\alpha}} L(\hat{\alpha}) \text{ subject to } \sum \alpha y^i = 0 \text{ and } \alpha \geq 0, \forall i$$

$$L(\hat{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i x_i x_j$$

- Has a unique maximum vector of $$\alpha_i$$
- Can be found by quadratic programming methods

To the Max

$$\max_{\hat{\alpha}} L(\hat{\alpha}) \text{ subject to } \sum \alpha y^i = 0 \text{ and } \alpha \geq 0, \forall i$$

$$L(\hat{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i x_i x_j$$

- Has a unique minimum vector of $$\alpha_i$$
- Can be found by quadratic programming methods

Gradient Ascent for Max Margin “Adatron”

- $$\alpha = 0$$
- Repeat until the $$\alpha_i$$ stop changing
- Repeat for each point i

$$\alpha_i \leftarrow \alpha_i + \eta \left(1 - y_i \sum \alpha_i y_i x_i x_i \right)$$

- If $$\alpha_i < 0$$ then $$\alpha_i \leftarrow 0$$

$$\alpha_i$$, $$x_i$$, $$y_i$$ and $$y_i$$ are from training data, thus known.
Simple SVM Problem
Formulation

\[ \begin{align*}
\min_{w_1, w_2} & \quad \frac{1}{2} (w_1^2 + w_2^2) \\
\text{subject to} & \quad -1(w_1^2 + w_2^2 + b) - 1 \geq 0 \\
& \quad 1(w_1^2 + w_2^2 + b) - 1 \geq 0
\end{align*} \]

Simple SVM Problem
Lagrangian

Define Lagrangian \( L(w) \)

\[ L(w) = \frac{1}{2} (w_1^2 + w_2^2) - \alpha_1(-1 - b) - \alpha_2(w_1 + w_2 + b - 1) \]

Maximize Lagrangian \( L(w) \) with respect to \( w \) and \( b \)

\[ \frac{\partial L(w)}{\partial w_1} = w_1 - \alpha_2 = 0 \]
\[ \frac{\partial L(w)}{\partial w_2} = w_2 - \alpha_2 = 0 \]
\[ \frac{\partial L(w)}{\partial b} = \alpha_1 - \alpha_2 = 0 \]

Simple SVM Problem
Weights and Offset

Both data points are support vectors, since \( \alpha_i > 0 \)

\[ w = \sum_{\text{support vectors}} \alpha_i y_i x_i \]

Use \( b = -1 \)

Simple SVM Problem
Lagrangian Dual

\[ \begin{align*}
w_1 &= \alpha_2 \\
w_2 &= \alpha_2 \\
\alpha_1 &= \alpha_2
\end{align*} \]

Maximize Lagrangian Dual \( L(\alpha) \) with respect to \( \alpha \)

\[ \frac{\partial L(\alpha)}{\partial \alpha_i} = 2 - 2\alpha_i = 0 \]

Simple SVM Problem
Dual

\[ \begin{align*}
\min_{\alpha_i} & \quad \frac{1}{2} (\alpha^2 + \alpha^2) \\
\text{subject to} & \quad -1(\alpha_1 - \alpha_2) - \alpha_2(\alpha_1 + \alpha_2 + b - 1) \\
& \quad (\alpha_1 + \alpha_2) - \alpha_2^2 + b(\alpha_1 - \alpha_2) \quad \text{use } \alpha_i = \alpha_i
\end{align*} \]

It satisfies constraints \( \alpha_i \geq 0 \) and \( \sum \alpha_i = 0 \)

Both data points are support vectors, since \( \alpha_i > 0 \)

\[ w = \sum_{\text{support vectors}} \alpha_i y_i x_i \]

\[ b = -1 \]