Support Vector Machines
Outline

• **Linear SVMs**
• The definition of a maximum margin classifier
• What QP
• How Maximum Margin can be turned into a QP problem
• How we deal with noisy (non-separable) data
• How we permit non-linear boundaries
Linear Classifiers

How would you classify this data?

Learning machine $f$ takes an input $x$ and transforms it, somehow using weight $\alpha$ into a predicted output $y^\text{est} = +/-1$
Linear Classifiers

\[ f(x, w, b) = \text{sign}(w \cdot x - b) \]

- denotes +1
- denotes -1

Perceptron rule stops only if no misclassification remains:

\[ \Delta w = \eta (t - y) x \]

→ Any of these lines are possible
Linear Classifiers

\[ f(x, w, b) = \text{sign}(w \cdot x - b) \]

- denotes +1
- denotes -1

If the objective is to assure zero sample error...

Any of these would be fine..

..but which is best?
Classifier Margin

- denotes +1
- denotes -1

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Margin of the separator is the width of separation between classes.

\[ f(x, w, b) = \text{sign}(w \cdot x - b) \]
Maximum Margin

- denotes +1
- denotes -1

Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

\[
f(x,w,b) = \text{sign}(w \cdot x - b)
\]
Outline

- Linear SVMs
- **The definition of a maximum margin classifier**
- What QP
- How Maximum Margin can be turned into a QP problem
- How we deal with noisy (non-separable) data
- How we train SVMs
- How we permit non-linear boundaries
- How SVM Kernel functions permit
Specifying a line and margin

- How do we represent this mathematically?
- ...in $m$ input dimensions?
Specifying a line and margin

- Plus-plane: \( \{ x : w \cdot x + b = +1 \} \)
- Minus-plane: \( \{ x : w \cdot x + b = -1 \} \)

Classify as:

- \( +1 \) if \( w \cdot x + b \geq 1 \)
- \( -1 \) if \( w \cdot x + b \leq -1 \)
- \( X \) if \( -1 < w \cdot x + b < 1 \)
Computing the margin width

How do we compute $M$ in terms of $\mathbf{w}$ and $b$?

- Plus-plane = $\{ x : \mathbf{w} \cdot x + b = +1 \}$
- Minus-plane = $\{ x : \mathbf{w} \cdot x + b = -1 \}$

Claim: The vector $\mathbf{w}$ is perpendicular to the Plus Plane. Why?

Let $\mathbf{u}$ and $\mathbf{v}$ be two vectors on the Plus Plane. What is $\mathbf{w} \cdot (\mathbf{u} - \mathbf{v})$?

And so of course the vector $\mathbf{w}$ is also perpendicular to the Minus Plane.
Computing the margin width

The line from $x^-$ to $x^+$ is perpendicular to the planes. So to get from $x^-$ to $x^+$ travel some distance in direction $w$

- Plus-plane $= \{ x : w \cdot x + b = +1 \}$
- Minus-plane $= \{ x : w \cdot x + b = -1 \}$
- The vector $w$ is perpendicular to the Plus Plane
- Let $x^-$ be any point on the minus plane
- Let $x^+$ be the closest plus-plane-point to $x^-$. 
- Claim: $x^+ = x^- + \lambda \ w$ for some value of $\lambda$. Why?
Computing the margin width

What we know:

- \( \mathbf{w} \cdot \mathbf{x}^+ + b = +1 \)
- \( \mathbf{w} \cdot \mathbf{x}^- + b = -1 \)
- \( \mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w} \)
- \( |\mathbf{x}^+ - \mathbf{x}^-| = \mathcal{M} \)

It's now easy to get \( \mathcal{M} \) in terms of \( \mathbf{w} \) and \( b \)

\[
\mathbf{w} \cdot (\mathbf{x}^- + \lambda \mathbf{w}) + b = 1 \\
\Rightarrow \\
\mathbf{w} \cdot \mathbf{x}^- + b + \lambda \mathbf{w} \cdot \mathbf{w} = 1 \\
\Rightarrow \\
-1 + \lambda \mathbf{w} \cdot \mathbf{w} = 1 \\
\Rightarrow \\
\lambda = \frac{2}{\mathbf{w} \cdot \mathbf{w}}
\]
Computing the margin width

What we know:

- \( \mathbf{w} \cdot \mathbf{x}^+ + b = +1 \)
- \( \mathbf{w} \cdot \mathbf{x}^- + b = -1 \)
- \( \mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w} \)
- \( |\mathbf{x}^+ - \mathbf{x}^-| = M \)
- \( \lambda = \frac{2}{\mathbf{w} \cdot \mathbf{w}} \)

\[ M = |\mathbf{x}^+ - \mathbf{x}^-| = \lambda |\mathbf{w}| = \lambda \sqrt{\mathbf{w} \cdot \mathbf{w}} \]

\[ = \frac{2 \sqrt{\mathbf{w} \cdot \mathbf{w}}}{\mathbf{w} \cdot \mathbf{w}} = \frac{2}{\sqrt{\mathbf{w} \cdot \mathbf{w}}} \]
Outline

• Linear SVMs
• The definition of a maximum margin classifier
• **What QP**
• **How Maximum Margin can be turned into a QP problem**
• How we deal with noisy (non-separable) data
• How we train SVMs
• How we permit non-linear boundaries
• How SVM Kernel functions permit
Learning the Maximum Margin Classifier

Given a guess of $\mathbf{w}$ and $b$ we can

- Compute whether all data points in the correct half-planes
- Compute the width of the margin

So now we just need to write a program to search the space of $\mathbf{w}$’s and $b$’s to find the widest margin that matches all the datapoints. How?

Learning via Quadratic Programming

- QP is a well-studied class of optimization algorithms to maximize a quadratic function of some real-valued variables subject to linear constraints.

  - Quadratic optimization problem:

    | Find \( \mathbf{w} \) and \( b \) such that |
    | --- |
    | \[ \rho = \frac{2}{\|\mathbf{w}\|} \] is maximized and for all \( \{(\mathbf{x}_i, y_i)\} \) |
    | \( \mathbf{w}^T \mathbf{x}_i + b \geq 1 \) if \( y_i = 1 \); \( \mathbf{w}^T \mathbf{x}_i + b \leq -1 \) if \( y_i = -1 \) |

  - A better formulation:

    | Find \( \mathbf{w} \) and \( b \) such that |
    | --- |
    | \( \Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \) is minimized and for all \( \{(\mathbf{x}_i, y_i)\} \) |
    | \( y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \) |
The Optimization Problem

- The solution involves constructing a *dual problem* where a Lagrange multiplier $\alpha_i$ is associated with every constraint in the primary problem:

  Find $\alpha_1 \ldots \alpha_N$ such that
  
  $Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j x_i^T x_j$ is maximized and
  
  (1) $\sum \alpha_i y_i = 0$
  
  (2) $\alpha_i \geq 0$ for all $\alpha_i$

- The solution has the form:

  $w = \sum \alpha_i y_i x_i \quad b = y_k - w^T x_k$ for any $x_k$ such that $\alpha_k \neq 0$

- Each non-zero $\alpha_i$ indicates that corresponding $x_i$ is a support vector.

- Then the classifying function will have the form:

  $f(x) = \sum \alpha_i y_i x_i^T x + b$

- Notice that it relies on an *inner product* between the test point $x$ and the support vectors $x_i$.

- Also keep in mind that solving the optimization problem involved computing the inner products $x_i^T x_j$ between all training points!
Learning the Maximum Margin Classifier

The objective?
- To maximize M
- While ensuring all data points are in the correct half-planes

Q1: What is our quadratic optimization criterion? Minimize $w \cdot w$

Q2: What are constraints in our QP?
- How many constraints will we have? $R$
- What should they be?
  - Constraints:
    
    
    $w \cdot x_k + b \geq 1$ if $y_k = 1$
    $w \cdot x_k + b \leq -1$ if $y_k = -1$
maximize $f(x, y)$

subject to $g(x, y) = c$.

$L(x, y, \lambda) = f(x, y) + \lambda \cdot (g(x, y) - c)$,

Solve: $\nabla_{x,y,\lambda} L(x, y, \lambda) = 0$.

$\nabla_{x,y} f = -\lambda \nabla_{x,y} g$

$\nabla_{\lambda} L(x, y, \lambda) = 0 \Rightarrow g(x, y) = c$
Karush-kuhn-Tucker condition

Minimize \( f(x) \)

subject to:

\[ g_i(x) \leq 0, \quad h_j(x) = 0 \]

\( g_i(\cdot) (i = 1, \ldots, m) \quad h_j(\cdot) (j = 1, \ldots, l) \)

KKT multiplier: \( L_p = f(x) + \sum_{i=1}^{m} \mu_i g_i(x) + \sum_{j=1}^{l} \lambda_j h_j(x) \)

\[ \nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{l} \lambda_j \nabla h_j(x^*) = 0, \]

\[ g_i(x^*) \leq 0, \text{ for all } i = 1, \ldots, m \quad \text{: Primal feasibility} \]

\[ h_j(x^*) = 0, \text{ for all } j = 1, \ldots, l \]

\[ \mu_i \geq 0, \text{ for all } i = 1, \ldots, m \quad \text{: Dual feasibility} \]

\[ \mu_i g_i(x^*) = 0, \text{ for all } i = 1, \ldots, m. \quad \text{: Complementary slackness} \]
SVM – Hard Margin

\[
\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{L} \xi_i \quad \text{s.t.} \quad y_i (x_i \cdot w + b) - 1 + \xi_i \geq 0 \quad \forall i
\]

KKT condition

\[
L_P \equiv \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{L} \xi_i - \sum_{i=1}^{L} \alpha_i [y_i (x_i \cdot w + b) - 1 + \xi_i] - \sum_{i=1}^{L} \mu_i \xi_i
\]

\[
\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L} \alpha_i y_i x_i
\]

\[
\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0
\]

\[
\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow \mu_i = \alpha_i
\]

\[
L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j
\]

Tool: SVM-light, LIBSVM (SMO algorithm)
Outline

- Linear SVMs
- The definition of a maximum margin classifier
- What QP
- How Maximum Margin can be turned into a QP problem
- **How we deal with noisy (non-separable) data**
- How we train SVMs
- How we permit non-linear boundaries
- How SVM Kernel functions permit
Uh-oh!

This is going to be a problem!
What should we do?

Idea:

Minimize $w.w + C$ (distance of error points to their correct place)

Minimize trade-off between margin and training error
Learning Maximum Margin with Noise

Our QP criterion:
Minimize
\[ \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^{R} \varepsilon_k \]

- How many constraints? \(2R\)
- What should they be?
  \[ \mathbf{w} \cdot \mathbf{x}_k + b \geq 1 - \varepsilon_k \text{ if } y_k = 1 \]
  \[ \mathbf{w} \cdot \mathbf{x}_k + b \leq -1 + \varepsilon_k \text{ if } y_k = -1 \]
  \[ \varepsilon_k \geq 0 \text{ for all } k \]
**Hard/Soft Margin Separation**

**Idea:** Maximize margin and minimize training error simultaneously.

**Hard Margin:**

\[
\text{minimize } J(\hat{w}, b) = \frac{1}{2} \hat{w} \cdot \hat{w}
\]

\[\text{s. t. } y_i[\hat{w} \cdot \hat{x}_i + b] \geq 1\]

**Soft Margin:**

\[
\text{minimize } J(\hat{w}, b, \xi) = \frac{1}{2} \hat{w} \cdot \hat{w} + C \sum_{i=1}^{n} \xi_i
\]

\[\text{s. t. } y_i[\hat{w} \cdot \hat{x}_i + b] \geq 1 - \xi_i \text{ and } \xi_i \geq 0\]

- slack variable $\xi_i$ measures by how much example $(x_i, y_i)$ fails to achieve a target margin of $\delta$.
- $\sum \xi_i$ is an upper bound on the number of training errors.
- $C$ is a parameter that controls trade-off between margin and error.
Controlling Soft Margin Separation

Soft Margin: minimize $P(w, b, \xi) = \frac{1}{2} \vec{w} \cdot \vec{w} + C \sum_{i=1}^{n} \xi_i$

s. t. $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

- $\sum \xi_i$ is an upper bound on the number of training errors.
- $C$ is a parameter that controls trade-off between margin and error.

Copyright © 2001, Andrew W. Moore
SVM – Soft Margin

\[
\begin{align*}
\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{L} \xi_i & \quad \text{s.t.} \quad y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0 \quad \forall i \\
L_P \equiv \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{L} \xi_i - \sum_{i=1}^{L} \alpha_i [y_i(x_i \cdot w + b) - 1 + \xi_i] - \sum_{i=1}^{L} \mu_i \xi_i
\end{align*}
\]

\[
\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L} \alpha_i y_i x_i
\]

\[
\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0
\]

\[
\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow C = \alpha_i + \mu_i \quad \Rightarrow \quad 0 \leq \alpha_i \leq C, \forall i
\]

\[
L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j
\]

Tool: SVM-light, LIBSVM (SMO algorithm)
min \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{L} \xi_i \quad \text{s.t.} \quad y_i (\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i \geq 0 \quad \forall i

\text{KKT condition}

L_P \equiv \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{L} \xi_i - \sum_{i=1}^{L} \alpha_i [y_i (\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i] - \sum_{i=1}^{L} \nu_i \xi_i

\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i

\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0

\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow \nu_i = \alpha_i + \mu_i

L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j

\text{Tool: SVM-light, LIBSVM (SMO algorithm)}

Support Vector Machines: Slide 29
Outline

• Linear SVMs
• The definition of a maximum margin classifier
• What QP
• How Maximum Margin can be turned into a QP problem
• How we deal with noisy (non-separable) data
• How we permit non-linear boundaries
• How SVM Kernel functions permit
Outline

- Linear SVMs
- The definition of a maximum margin classifier
- What QP
- How Maximum Margin can be turned into a QP problem
- How we deal with noisy (non-separable) data
- **How we permit non-linear boundaries**
- How SVM Kernel functions permit
Suppose we’re in 1-dimensional space.

What would SVMs do with this data?
Suppose we’re in 1-dimension

Not a big surprise
Harder 1-dimensional dataset

That’s wiped the smirk off SVM’s face.

What can be done about this?

\[ x = 0 \]
Harder 1-dimensional dataset

Remember how permitting non-linear basis functions made linear regression so much nicer?

Let’s permit them here too

\[ z_k = (x_k, x_k^2) \]
Harder 1-dimensional dataset

Remember how permitting non-linear basis functions made linear regression so much nicer?

Let’s permit them here too

\[ z_k = (x_k, x_k^2) \]
Outline

- Linear SVMs
- The definition of a maximum margin classifier
- What QP
- How Maximum Margin can be turned into a QP problem
- How we deal with noisy (non-separable) data
- How we permit non-linear boundaries
Doing multi-class classification

- SVMs can only handle two-class outputs (i.e. a categorical output variable with arity 2).
- What can be done?
- Answer: with output arity N, learn N SVM’s
  - SVM 1 learns “Output==1” vs “Output != 1”
  - SVM 2 learns “Output==2” vs “Output != 2”
  - ...
  - SVM N learns “Output==N” vs “Output != N”
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.
**SVMlight**

- [http://svmlight.joachims.org](http://svmlight.joachims.org) (Thorsten Joachims @ cornell Univ.)

- **Commands**
  - `svm_learn [options] [train_file] [model_file]`
    - `svm_learn --c 1.5 --x 1 train_file model_file`
  - `svm_classify [test_file] [model_file] [prediction_file]`
    - `svm_classify test_file model_file prediction_file`

- **File Format**
  - `<y> <featureNumber>:<value> ... <featureNumber>:<value>`
    #comment
  - `1 23:0.5 105:0.1 1023:0.8 1999:0.34`

| Idx | 1   | ... | 23  | ... | 105 | ... | 1023 | 1024 | ... | 1998 | 1999 | ... | cls |
|-----|-----|-----|-----|-----|-----|-----|------|------|-----|------|------|-----|-----|-----|
| Value | 0   | 0   | .5  | 0   | .1  | 0   | .8   | 0    | 0   | 0    | .34  | 0   | 1   |
Options

- \(-z \ <\text{char}\) : selects whether the SVM is trained in classification (c) or regression (r) mode
- \(-c \ <\text{float}\) : controls the trade-off between training error and margin – the lower the value of C, the more training error is tolerated. The best value of C depends on the data and must be determined empirically.
- \(-w \ <\text{float}\) : width of tube (i.e \(\epsilon\)) \(\epsilon\)-intensive loss function used in regression. The value must be non-negative.
- \(-j \ <\text{float}\) : specifies cost-factors for the loss functions both in classification and regression.
- \(-b \ <\text{int}\) : switches between a biased (1) or an unbiased (0) hyperplane.
- \(-i \ <\text{int}\) : selects how training errors are treated. (1-4)
- \(-x \ <\text{int}\) : selects whether to compute leave-one-out estimates of the generalization performance.
- \(-t \ <\text{int}\) : selects the type of kernel functions (0-4)
- \(-q \ <\text{int}\) : maximum size of working set.
- \(-m \ <\text{int}\) : specifies the amount of memory (in megabytes) available for caching kernel evaluations.

..................................................
Example 1: Text Chunking

- **Corpus**
  - [http://sejong.knu.ac.kr/~sbpark/Chunk](http://sejong.knu.ac.kr/~sbpark/Chunk)

<table>
<thead>
<tr>
<th>Information</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vocabulary</td>
<td>16,838</td>
</tr>
<tr>
<td>Total Words</td>
<td>321,328</td>
</tr>
<tr>
<td>Chunk Types</td>
<td>9</td>
</tr>
<tr>
<td>POS Tags</td>
<td>52</td>
</tr>
<tr>
<td>Sentences</td>
<td>12,092</td>
</tr>
<tr>
<td>Phrases</td>
<td>112,658</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word</th>
<th>Tag</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>또한</td>
<td>maj</td>
<td>B-ADVP</td>
</tr>
<tr>
<td>그</td>
<td>mmd</td>
<td>B-NP</td>
</tr>
<tr>
<td>사실</td>
<td>ncn</td>
<td>I-NP</td>
</tr>
<tr>
<td>은</td>
<td>jxt</td>
<td>I-NP</td>
</tr>
<tr>
<td>기지</td>
<td>ncn</td>
<td>B-NP</td>
</tr>
<tr>
<td>의</td>
<td>jcm</td>
<td>I-NP</td>
</tr>
<tr>
<td>안전</td>
<td>ncps</td>
<td>I-NP</td>
</tr>
<tr>
<td>예</td>
<td>jca</td>
<td>I-NP</td>
</tr>
<tr>
<td>매우</td>
<td>mag</td>
<td>B-ADVP</td>
</tr>
<tr>
<td>좋다</td>
<td>paa</td>
<td>B-VP</td>
</tr>
<tr>
<td>을 수</td>
<td>ef</td>
<td>I-VP</td>
</tr>
<tr>
<td>있 다</td>
<td>nbn</td>
<td>I-VP</td>
</tr>
<tr>
<td>da</td>
<td>paa</td>
<td>I-VP</td>
</tr>
<tr>
<td>.</td>
<td>ef</td>
<td>I-VP</td>
</tr>
<tr>
<td>.</td>
<td>sf</td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decision Tree</th>
<th>SVM</th>
<th>MBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>97.95±0.24%</td>
<td>98.15±0.20%</td>
</tr>
<tr>
<td>F-score</td>
<td>91.36±0.85</td>
<td>92.54±0.72</td>
</tr>
</tbody>
</table>
Example 2: Text Classification

- Corpus: Reuters-21578
  - 12,902 news story
  - 118 category
  - ModApte split (75% training: 9,603 docs / 25% testing 3,299 docs)

![Graph showing precision-recall curve for different classifiers](image)