SVMs: Linear and Beyond

Advanced Statistical Methods in NLP
Ling 572
February 21, 2012
SVM Training

- Goal: Maximum margin, consistent w/training data

\[ d = \frac{1}{||w||} \]

\[ <w,x>+b=-1 \]

\[ <w,x>+b=0 \]

Fig. from F. Xia
SVM Training

- Goal: Maximum margin, consistent w/ training data
  - Margin = 2d = 2/||w||

Fig. from F. Xia
SVM Training

- Goal: Maximum margin, consistent w/training data
  - Margin = 2d = 2/||w||
- How can we maximize?

Fig. from F. Xia
SVM Training

- Goal: Maximum margin, consistent with training data
  - Margin = $2d = 2/||w||$
- How can we maximize?
  - Max $d$ →

![Diagram showing SVM training with margin and support vectors]

Fig. from F. Xia
SVM Training

- Goal: Maximum margin, consistent with training data
  - Margin = 2d = 2/||w||

- How can we maximize?
  - Max d $\rightarrow$ Min ||w||

Fig. from F. Xia
SVM Training

- Goal: Maximum margin, consistent w/training data
- So we will:
  - Minimize $||w||^2$
  - subject to

Fig. from F. Xia
SVM Training

- Goal: Maximum margin, consistent w/training data

- So we will:
  - Minimize $||w||^2$
  - subject to $y_i(<w,x_i>+b) \geq 1$

Fig. from F. Xia
SVM Training

- Goal: Maximum margin, consistent w/training data

- So we will:
  - Minimize $||w||^2$
    - subject to
  - $y_i(<w,x_i>+b) >= 1$
  - $y_i(<w,x_i>+b)-1 >= 0$

Fig. from F. Xia
SVM Training

- Goal: Maximum margin, consistent w/training data

- So we will:
  - Minimize $||w||^2$
    - subject to
  - $y_i(<w,x_i>+b) \geq 1$
  - $y_i(<w,x_i>+b)-1 \geq 0$

- Quadratic programming (QP) problem
  - Can use standard packages to solve

Fig. from F. Xia
Lagrangian Conversion

- Have constrained optimization problem
- Convert to unconstrained optimization
  - Using Lagrange multipliers
Lagrangian Conversion

- Have constrained optimization problem
  - Convert to unconstrained optimization
    - Using Lagrange multipliers
- Minimize $||w||^2$ subject to $y_i(<w,x_i>+b)-1 \geq 0$
Lagrangian Conversion

- Have constrained optimization problem
- Convert to unconstrained optimization
  - Using Lagrange multipliers

- Minimize $||w||^2$ subject to $y_i(<w,x_i>+b)-1 >=0$

- $L(\bar{w}, b, \alpha) = \frac{1}{2} ||w||^2 - \sum_i \alpha_i (y_i(\bar{w}, \bar{x}) + b) - 1)$
Lagrangian Conversion

- Have constrained optimization problem
- Convert to unconstrained optimization
  - Using Lagrange multipliers

- Minimize $||w||^2$ subject to $y_i(<w,x_i>+b)-1 \geq 0$
- $L(\vec{w},b,\alpha) = \frac{1}{2}||w||^2 - \sum_i \alpha_i(y_i(<\vec{w},\vec{x}_i>+b)-1)$
- $\vec{w} = \sum_{i=1}^N \alpha_i y_i \vec{x}_i$ and $\sum_{i=1}^N \alpha_i y_i = 0$
Decoding

- Given $w, b$, predict label
Decoding

- Given $w, b$, predict label
- Prediction:
  - $f(x) = \text{sign} \ (<w, x> + b)$
Decoding

- Given $w, b$, predict label

Prediction:
- $f(x) = \text{sign} \ (<w, x> + b)$

Suppose $w = (2, -3), b = 3$
- $x = (3, 1)$
Decoding

- Given $w, b$, predict label
- Prediction:
  - $f(x) = \text{sign} \ (<w,x>+b)$
- Suppose $w=(2,-3), b=3$
- $x=(3,1) \Rightarrow (6+(-3))+3=6 \Rightarrow +$
- $x=(-1,1)$
Decoding

- Given \( w, b \), predict label
- Prediction:
  - \( f(x) = \text{sign} \left( <w,x> + b \right) \)
- Suppose \( w=(2,-3), \ b=3 \)
  - \( x=(3,1) \implies (6+(-3))+3=6 \implies + \)
  - \( x=(-1,1) \implies (-2-3)+3 = -2 \implies _{-} \)
Decoding

- Given $w, b$, predict label
- Prediction:
  - $f(x) = \text{sign} \ (<w,x>+b)$

- What if the point is ‘in the margin’?
  - Just classify by sign regardless
  - Return “don’t know”
Using the Trained Values

- Training: Learns $\alpha_i$
- Use those to compute $w$
Using the Trained Values

- Training: Learns $\alpha_i$
- Use those to compute $w$

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$$

$x_1=(3,0,-1); y_1=1; \alpha_1 = 3$
$x_2=(-1,2,0); y_2=-1; \alpha_2 = 1$
$x_3=(-5,4,-7); y_3=-1; \alpha_3 = 0$
Using the Trained Values

- Training: Learns $\alpha_i$
- Use those to compute $w$

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$$

$x_1= (3,0,-1); y_1=1; \alpha_1 = 3$
$x_2= (-1,2,0); y_2=-1; \alpha_2 = 1$
$x_3= (-5,4,-7); y_3=-1; \alpha_3 = 0$

- $w= (9,0,-3)+(1,-2,0)$
- $= (10,-2,-3)$
Using Trained Values

- Decoding:
  - \( f(x) = \text{sign}(<w,x>+b) \)
Using Trained Values

- Decoding:
- \( f(x) = \text{sign}(\langle w,x \rangle + b) \)

\[
f(x) = \text{sign} \left( \sum_{i=1}^{n} \alpha_i y_i x_i \bar{x} + b \right)
\]
Using Trained Values

- Decoding:
  - \( f(x) = \text{sign}(\langle w, x \rangle + b) \)

\[
\begin{align*}
    f(x) & = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i \bar{x}_i \bar{x} + b) \\
    & = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i (\langle \bar{x}_i, \bar{x} \rangle + b))
\end{align*}
\]
Trained Values

- Training learns $\alpha_i$
- For support vectors,
Trained Values

- Training learns $\alpha_i$
- For support vectors, $\alpha_i > 0$
- For all other training instances
Trained Values

- Training learns $\alpha_i$
- For support vectors, $\alpha_i > 0$
- For all other training instances, $\alpha_i = 0$
  - Non-support vectors ignored
Trained Values

- Training learns $\alpha_i$
- For support vectors, $\alpha_i > 0$
- For all other training instances, $\alpha_i = 0$
  - Non-support vectors ignored

- Training learns:
  - To identify support vectors and set $\alpha_i$ values
    - Equivalent to computing weights
      - Avoids computing weights which can be problematic
Basic SVMs

- Training:
  - Minimize $||w||^2$ subject to $<w,x>+b=1$
Basic SVMs

- **Training:**
  - Minimize $||w||^2$ subject to $<w,x>+b=1$

- **Decoding:**
  - Compute $f(x)=\text{sign} (<w,x>+b)$
Basic SVMs (Lagrangian)

- Training:
  -
Basic SVMs (Lagrangian)

- Training:
  - Minimize: \( L(\tilde{w}, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i (y_i (\langle \tilde{w}, \tilde{x} \rangle + b) - 1) \)
  - w.r.t. w, b

- Decoding:
Basic SVMs (Lagrangian)

- **Training:**
  - Minimize: \( L(\tilde{w}, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i(y_i(<\tilde{w}, \tilde{x}> + b) - 1) \)
  - w.r.t. \( w, b \)

- **Decoding:**
  - Compute
    \[
    f(x) = \text{sign}(\sum_{i=1}^n \alpha_i y_i \tilde{x}_i \tilde{x} + b)
    \]
    \[
    = \text{sign}(\sum_{i=1}^n \alpha_i y_i(<\tilde{x}_i, \tilde{x}> + b))
    \]
kNN vs SVM

- Voting in kNN:
  - Majority vote: \( c^* = \arg\max_c g(c) \)
kNN vs SVM

- Voting in kNN:
  - Majority vote: $c^* = \arg\max_c g(c)$
  - Weighted voting: $c^* = \arg\max_c \sum_i w_i \delta(c, f_i(x))$

Due to F. Xia
kNN vs SVM

- Voting in kNN:
  - Majority vote: $c^* = \text{argmax}_c g(c)$
  - Weighted voting: $c^* = \text{argmax}_c \sum_i w_i \delta(c, f_i(x))$

- Weighted voting allows use of many training examples
  - $w_i = 1/\text{dist}(x, x_i)$
  - Could use all training examples

Due to F. Xia
kNN vs SVM

- Voting in kNN:
  - Majority vote: \( c^* = \text{argmax}_c \ g(c) \)
  
- Weighted voting: \( c^* = \text{argmax}_c \ \sum_i w_i \delta(c, f_i(x)) \)

- Weighted voting allows use of many training examples
  - \( w_i = 1 / \text{dist}(x, x_i) \)
  - Could use all training examples
  - \( f(\tilde{x}) = \sum_i w_i y_i \) binary case
kNN vs SVM

- Voting in kNN:
  - Majority vote: \( c^* = \text{argmax}_c \ g(c) \)
  - Weighted voting: \( c^* = \text{argmax}_c \ \sum_i w_i \delta(c,f_i(x)) \)

- Weighted voting allows use of many training examples
  - \( w_i = 1 / \text{dist}(x,x_i) \)
  - Could use all training examples
  - SVM: \( f(x) = \sum_i \alpha_i y_i < x_i, x > + b \)
  - Weighted voting allows use of many training examples
Summary

- Support Vector Machines:
  - Find decision hyperplane $\langle w, x \rangle + b = 0$
  - Among possible hyperplanes, select one that maximizes margin
  - Maximizing margin equivalent to minimizing $||w||$
  - Solve by learning alphas for each training instance
    - Non-zero only for support vectors
Beyond Linear SVMs

- So far, we’ve assumed data is linearly separable
  - Some margin exists b/t samples of different classes
Beyond Linear SVMs

- So far, we’ve assumed data is linearly separable
  - Some margin exists b/t samples of different classes

- Problem:
  - Not all data/problems are linearly separable
Beyond Linear SVMs

- So far, we’ve assumed data is linearly separable
  - Some margin exists b/t samples of different classes

- Problem:
  - Not all data/problems are linearly separable

- Two variants:
  - Data intrinsically linearly separable
    - But noisy data points appear in margin or misclassified
    - ➔ Soft-margins
Beyond Linear SVMs

- So far, we’ve assumed data is linearly separable
  - Some margin exists b/t samples of different classes

- Problem:
  - Not all data/problems are linearly separable

- Two variants:
  - Data intrinsically linearly separable
    - But noisy data points appear in margin or misclassified
      - Soft-margins
  - Data really isn’t linearly separable
    - Non-linear SVMs
Soft–Margin Intuition

- High dimensional data (i.e. text classification docs)
Soft–Margin Intuition

- High dimensional data (i.e. text classification docs)
- Often not linearly separable
  - E.g. due to noise or data
Soft–Margin Intuition

- High dimensional data (i.e. text classification docs)
  - Often not linearly separable
    - E.g. due to noise or data
  - Prefer simpler linear model
Soft–Margin Intuition

- High dimensional data (i.e. text classification docs)
  - Often not linearly separable
    - E.g. due to noise or data
    - Prefer simpler linear model

- Allow soft-margin classification
  - Margin decision boundary can make some mistakes
    - Points inside margin or wrong side of boundary
Soft–Margin Intuition

- High dimensional data (i.e. text classification docs)
  - Often not linearly separable
    - E.g. due to noise or data
  - Prefer simpler linear model

- Allow soft-margin classification
  - Margin decision boundary can make some mistakes
    - Points inside margin or wrong side of boundary
  - Pay a penalty for each such item
    - Dependent on how far it is from required margin
Implementation

- Modify our optimization criteria
Implementation

- Modify our optimization criteria
  - Two aspects
    - Modify constraints to allow instances to be ‘off by a bit’
Implementation

• Modify our optimization criteria
  • Two aspects
    • Modify constraints to allow instances to be ‘off by a bit’
    • Modify optimization term to incorporate error penalty
Implementation

- Modify our optimization criteria
  - Two aspects
    - Modify constraints to allow instances to be ‘off by a bit’
  - Modify optimization term to incorporate error penalty
- Mechanism: Introduce ‘slack variables’
  - Incorporate in margin constraints
Implementation

- Modify our optimization criteria
  - Two aspects
    - Modify constraints to allow instances to be ‘off by a bit’
  - Modify optimization term to incorporate error penalty

- Mechanism: Introduce ‘slack variables’
  - Incorporate in margin constraints
  - Incorporate in optimization target
Soft-Margin

- Minimize

$$\frac{1}{2} \|w\|^2 + C \sum_i \xi_i$$
Soft-Margin

• Minimize

$$\frac{1}{2} \|w\|^2 + C \sum_{i} \xi_i$$

• Subject to

$$y_i (\langle \bar{w}, \bar{x}_i \rangle + b) \geq 1 - \xi_i$$
Soft-Margin

- Minimize

\[
\frac{1}{2} \|w\|^2 + C \sum \xi_i
\]

- Subject to

\[
y_i (\langle \bar{w}, \bar{x}_i \rangle + b) \geq 1 - \xi_i
\]

- \(C\) is a regularization term: controls overfitting
  - Trades off training error vs margin width
Soft-Margin

- Minimize

\[ \frac{1}{2} \| w \|^2 + C \sum_i \xi_i \]

- Subject to

\[ y_i (\langle \bar{w}, \bar{x}_i \rangle + b) \geq 1 - \xi_i \]

- C is a regularization term: controls overfitting
  - Trades off training error vs margin width

- \( \xi_i \geq 0 \)
Dual Problem Form

- Maximize:

\[
\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j < \bar{x}_i, \bar{x}_j >
\]
Dual Problem Form

- Maximize:

\[ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j < \bar{x}_i, \bar{x}_j > \]

- where \( 0 \leq \alpha_i \leq C, \sum_i \alpha_i y_i = 0 \)
Dual Problem Form

- Maximize:
  \[ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j < \bar{x}_i, \bar{x}_j > \]

- where \( 0 \leq \alpha_i \leq C, \sum_i \alpha_i y_i = 0 \)

- Note that the slack variables drop out & C bounds
Dual Problem Form

- Maximize:

\[ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j < \bar{x}_i, \bar{x}_j > \]

- where \( 0 \leq \alpha_i \leq C, \sum_i \alpha_i y_i = 0 \)

- Note that the slack variables drop out & C bounds

- Yields same form for w as before
  - b value includes \( \xi \) factor
Non-Linear SVMs

- Problem:
  - Sometimes data really isn’t linearly separable
Non-Linear SVMs

- Problem:
  - Sometimes data really isn’t linearly separable

- Approach:
  - Map data non-linearly into higher dimensional space
    - Data is separable in the higher dimensional space
Non-Linear SVMs

Problem:
- Sometimes data really isn’t linearly separable

Approach:
- Map data non-linearly into higher dimensional space
  - Data is separable in the higher dimensional space

Figure from Hearst et al ‘98
Feature Space

- Basic approach:
  - Original data is not linearly separable
Feature Space

- Basic approach:
  - Original data is not linearly separable

- Map data into ‘feature space’
  - Higher dimensional dot product space
  - Mapping via non-linear map: $\Phi$
Feature Space

• Basic approach:
  • Original data is not linearly separable
  
  • Map data into ‘feature space’
    • Higher dimensional dot product space
    • Mapping via non-linear map: $\Phi$

• Compute separating hyperplane
  • In higher dimensional space
Issues with Feature Space

- Mapping idea is simple,
- But has some practical problems
Issues with Feature Space

- Mapping idea is simple,
  - But has some practical problems

- Feature space can be very high – infinite? – dimensional
Issues with Feature Space

- Mapping idea is simple,
  - But has some practical problems

- Feature space can be very high – infinite? – dimensional

- Approach depends on computing similarity (dot product)
  - Computationally expensive
Issues with Feature Space

- Mapping idea is simple,
  - But has some practical problems

- Feature space can be very high – infinite? – dimensional

- Approach depends on computing similarity (dot product)
  - Computationally expensive

- Approach depends on mapping:
  - Also possibly intractable to compute
Solution

- “Kernel trick”:
- Use a kernel function $K: X \times X \rightarrow \mathbb{R}$

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$
Solution

• “Kernel trick”: Use a kernel function $K: X \times X \rightarrow R$
  
  
  \[ K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \]

• Computes similarity measure on images of data points
Solution

- “Kernel trick”:
  - Use a kernel function $K: X \times X \rightarrow R$
    
    $$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$
  
  - Computes similarity measure on images of data points
  - Can often compute similarity efficiently even on high (or infinite) dimensional space
Solution

• “Kernel trick”:
  • Use a kernel function $K: X \times X \rightarrow R$

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

• Computes similarity measure on images of data points
  • Can often compute similarity efficiently even on high (or infinite) dimensional space

• Choice of $K$ equivalent to selection of $\Phi$
Figure 20.27  (a) A two-dimensional training with positive examples as black circles and negative examples as white circles. The true decision boundary, $x_1^2 + x_2^2 \leq 1$, is also shown. (b) The same data after mapping into a three-dimensional input space $(x_1^2, x_2^2, \sqrt{2}x_1 x_2)$. The circular decision boundary in (a) becomes a linear decision boundary in three dimensions.
Example (cont’d)

- Original 2-D data: \( \mathbf{x} = (x_1, x_2) \)
Example (cont’d)

- Original 2-D data: \( \mathbf{x} = (x_1, x_2) \)
- Mapping to new values in 3-D feature space \( F(\mathbf{x}) \):
  - \( f_1 = x_1^2 \); \( f_2 = x_2^2 \); \( f_3 = \sqrt{2} x_1 x_2 \)
Example (cont’d)

- Original 2-D data: \( \mathbf{x}=(x_1, x_2) \)
- Mapping to new values in 3-D feature space \( F(x) \):
  - \( f_1 = x_1^2; \) \( f_2 = x_2^2; \) \( f_3 = \sqrt{2}x_1x_2 \)

\[
\bar{x} = (1, 2); \quad \bar{z} = (-2, 3)
\]
Example (cont’d)

- Original 2-D data: \( \mathbf{x} = (x_1, x_2) \)
- Mapping to new values in 3-D feature space \( F(x) \):
  - \( f_1 = x_1^2; \quad f_2 = x_2^2; \quad f_3 = \sqrt{2}x_1x_2 \)

\[
\tilde{x} = (1, 2); \quad \tilde{z} = (-2, 3)
\]

\[
\phi(\tilde{x}) =
\]
Example (cont’d)

- Original 2-D data: \( \mathbf{x} = (x_1, x_2) \)
- Mapping to new values in 3-D feature space \( \mathbf{F}(\mathbf{x}) \):
  - \( f_1 = x_1^2; \quad f_2 = x_2^2; \quad f_3 = \sqrt{2}x_1x_2 \)

\[ \mathbf{x} = (1, 2); \quad \mathbf{z} = (-2, 3) \]

\[ \phi(\mathbf{x}) = (1, 4, 2\sqrt{2}); \quad \phi(\mathbf{z}) = \]
Example (cont’d)

- Original 2-D data: \( \mathbf{x} = (x_1, x_2) \)

- Mapping to new values in 3-D feature space \( F(x) \):
  - \( f_1 = x_1^2 \); \( f_2 = x_2^2 \);
  - \( f_3 = \sqrt{2}x_1x_2 \)

\[
\tilde{x} = (1, 2); \quad \tilde{z} = (-2, 3)
\]

\[
\phi(\tilde{x}) = (1, 4, 2\sqrt{2}); \quad \phi(\tilde{z}) = (4, 9, -6\sqrt{2})
\]

\[
K(\tilde{x}, \tilde{z}) = \langle \phi(\tilde{x}), \phi(\tilde{z}) \rangle
\]

= 
Example (cont’d)

- Original 2-D data: \( x = (x_1, x_2) \)
- Mapping to new values in 3-D feature space \( F(x) \):
  - \( f_1 = x_1^2; \quad f_2 = x_2^2; \quad f_3 = \sqrt{2}x_1x_2 \)

\[
\begin{align*}
\vec{x} &= (1, 2); \quad \vec{z} = (-2, 3) \\
\phi(\vec{x}) &= (1, 4, 2\sqrt{2}); \quad \phi(\vec{z}) = (4, 9, -6\sqrt{2}) \\
K(\vec{x}, \vec{z}) &= \langle \phi(\vec{x}), \phi(\vec{z}) \rangle \\
&= 1 \cdot 4 + 4 \cdot 9 + 2\sqrt{2} \cdot -6\sqrt{2} \\
&= 16
\end{align*}
\]
Example (cont’d)

• More generally

\[ \tilde{x} = (x_1, x_2); \tilde{z} = (z_1, z_2) \]

\[ \phi(\tilde{x}) = \]
Example (cont’d)

- More generally

\[ \tilde{x} = (x_1, x_2); \tilde{z} = (z_1, z_2) \]

\[ \phi(\tilde{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2); \phi(\tilde{z}) = \]
Example (cont’d)

- More generally

\[ \vec{x} = (x_1, x_2); \vec{z} = (z_1, z_2) \]

\[ \phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2); \phi(\vec{z}) = (z_1^2, z_2^2, \sqrt{2}z_1z_2) \]

\[ \langle \phi(\vec{x}), \phi(\vec{z}) \rangle = \]
Example (cont’d)

- More generally

\[
\begin{align*}
\tilde{x} &= (x_1, x_2); \tilde{z} = (z_1, z_2) \\
\phi(\tilde{x}) &= (x_1^2, x_2^2, \sqrt{2}x_1x_2); \phi(\tilde{z}) = (z_1^2, z_2^2, \sqrt{2}z_1z_2) \\
\langle \phi(\tilde{x}), \phi(\tilde{z}) \rangle &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2
\end{align*}
\]
Example (cont’d)

- More generally

\[ \vec{x} = (x_1, x_2); \vec{z} = (z_1, z_2) \]
\[ \phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2); \phi(\vec{z}) = (z_1^2, z_2^2, \sqrt{2}z_1z_2) \]
\[ < \phi(\vec{x}), \phi(\vec{z}) >= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \]
\[ = (x_1z_1 + x_2z_2)^2 \]
Example (cont’d)

- More generally

\[ \vec{x} = (x_1, x_2); \quad \vec{z} = (z_1, z_2) \]

\[ \phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2); \quad \phi(\vec{z}) = (z_1^2, z_2^2, \sqrt{2}z_1z_2) \]

\[ < \phi(\vec{x}), \phi(\vec{z}) >= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \]

\[ = (x_1z_1 + x_2z_2)^2 \]

\[ = <x, z>^2 \]
Kernel Trick: Summary

- Avoids explicit mapping to high-dimensional space

- Avoids explicit computation of inner product in feature space

- Avoids explicit computation of mapping function
  - Or even feature vector

- Replace all inner products in SVM train/test with $K$
Non-Linear SVM Training

- Linear version:
  - Maximize
    \[ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j < \bar{x}_i, \bar{x}_j > \]
  - subject to
    \[ \alpha_i \geq 0; \sum_i \alpha_i y_i = 0 \]
Non-Linear SVM Training

• Linear version:
  • Maximize \( \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j < \bar{x}_i, \bar{x}_j > \)
  • subject to \( \alpha_i \geq 0; \sum_i \alpha_i y_i = 0 \)

• Non-linear version:
  • Maximize \( \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\bar{x}_i, \bar{x}_j) \)
Decoding

- Linear SVM:

\[ f(x) = \sum_i \alpha_i y_i < \tilde{x}_i, \bar{x} > + b \]
Decoding

- Linear SVM:

\[ f(x) = \sum_i \alpha_i y_i \langle \bar{x}_i, \bar{x} \rangle + b \]

- Non-linear SVM:

\[ f(x) = \sum_i \alpha_i y_i K(\bar{x}_i, \bar{x}) + b \]
Common Kernel Functions

- Implemented in most packages
- Linear: \( K(\mathbf{x}, \mathbf{z}) = <\mathbf{x}, \mathbf{z}> \)
- Polynomial: \( K(\mathbf{x}, \mathbf{z}) = (\gamma <\mathbf{x}, \mathbf{z}> + c)^d \)
- Radial Basis Function (RBF): \( K(\mathbf{x}, \mathbf{z}) = e^{-\gamma \|\mathbf{x}-\mathbf{z}\|^2} \)
- Sigmoid \( K(\mathbf{x}, \mathbf{z}) = \tanh(\gamma <\mathbf{x}, \mathbf{z}> + c) \)
- \( \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \)
Kernels

• Many conceivable kernels:
  • Function is a kernel if obeys Mercer’s theorem:
    • Is symmetric, continuous, and matrix is positive definite
Kernels

• Many conceivable kernels:
  • Function is a kernel if obeys Mercer’s theorem:
    • Is symmetric, continuous, and matrix is positive definite

• Selection of kernel can have huge impact
  • Dramatic differences in accuracy
Kernels

- Many conceivable kernels:
  - Function is a kernel if obeys Mercer’s theorem:
    - Is symmetric, continuous, and matrix is positive definite

- Selection of kernel can have huge impact
  - Dramatic differences in accuracy

- Knowledge about ‘shape’ of data can help select
  - Ironically, linear SVMs perform well on many tasks
Summary

- Find decision hyperplane that maximizes margin
Summary

- Find decision hyperplane that maximizes margin
- Employ soft-margin to support noisy data
Summary

- Find decision hyperplane that maximizes margin
- Employ soft-margin to support noisy data
- For non-linearly separable data, use non-linear SVMs
  - Project to higher dimensional space to separate
  - Use kernel trick to avoid intractable computation of
    - Projection or inner products
MaxEnt vs SVM

<table>
<thead>
<tr>
<th></th>
<th>MaxEnt</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling</td>
<td>Maximize $P(Y</td>
<td>X, \lambda)$</td>
</tr>
<tr>
<td>Training</td>
<td>Learn $\lambda_i$ for each feature function</td>
<td>Learn $\alpha_i$ for each training instance</td>
</tr>
<tr>
<td>Decoding</td>
<td>Calculate $P(y</td>
<td>x)$</td>
</tr>
<tr>
<td>Things to decode</td>
<td>Features Regularization Training algorithm</td>
<td>Kernel Regularization Training algorithm Binarization</td>
</tr>
</tbody>
</table>

Due to F. Xia