Instructor: Herke van Hoof (herke.vanhoof@mail.mcgill.ca)

Slides mostly by: Joelle Pineau

Class web page: www.cs.mcgill.ca/~hvanho2/comp551

Unless otherwise noted, all material posted for this course are copyright of the instructors, and cannot be reused or reposted without the instructors’ written permission.
Quizzes

Quizzes for:
- Instance based learning
- Support vector machines

Now available for self-test
Recap primal and dual

• Consider both solutions:

\[ p^* = \min_w \max_{\alpha: \alpha_i \geq 0} \, L(w, \alpha) \quad \text{Primal} \]
\[ d^* = \max_{\alpha: \alpha_i \geq 0} \, \min_w \, L(w, \alpha) \quad \text{Dual} \]

• The optimal \( w^*, \alpha^* \) are the same, as the following hold:
  – \( f \) and \( g_i \) are convex and
  – the \( g_i \) can all be satisfied simultaneously (for linearly seperable data)

• Thus we can choose whichever is easier to solve
  – Dual cubic in \( n \) (#examples)
  – Primal cubic in \( m \) (#features)
Recap primal and dual

• \( d^* = \max_{\alpha: \alpha_i \geq 0} \min_w L(w, \alpha) \) \hspace{1cm} \text{Dual}

• **Solution strategy: First step:** solve the inner problem \( \min_w L(w, \alpha) \):
  
  • Taking derivatives of \( L(w, \alpha) \) wrt \( w \), setting to 0, and solving for \( w \):
    \[
    L(w, \alpha) = \frac{1}{2} ||w||^2 + \sum_i \alpha_i (1 - y_i (w^T x_i)) \\
    \frac{\delta L}{\delta w} = w - \sum_i \alpha_i y_i x_i = 0 \\
    w^* = \sum_i \alpha_i y_i x_i
    \]

• Just like for the perceptron with zero initial weights, the optimal solution \( w^* \)
  is a linear combination of the \( x_i \).

• We can plug this in, and now find the \( \alpha \) that maximize the outer problem
Recap primal and dual

- **Result of first step:** The Lagrangian and solution for inner problem are:

  \[ d^* = \max_{\alpha : \alpha_i \geq 0} \min_w L(w, \alpha) \]

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

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

- **Solution strategy: Second step:** Plug this back into \( L \) to get the dual:

  \[ \max_{\alpha : \alpha_i \geq 0} L(w^*, \alpha) \]

  \[ \max_{\alpha : \alpha_i \geq 0} \frac{1}{2} ||w^*||^2 + \sum_i \alpha_i (1 - y_i (w^*^T x_i)) , \text{ use a bit of algebra to get} \]

  \[ \max_{\alpha : \alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (x_i^T x) \]

  with constraints \( \alpha_i \geq 0 \) and \( \sum_i \alpha_i y_i = 0 \).  

  Quadratic programming problem.
Comparison with logistic regression

• In logistic regression, the loss was:
  - $\log(\sigma(b+w^T x_i))$ for samples from the positive class
  - $\log(1 - \sigma(b+w^T x_i))$ for samples from the negative class

\[
\log(1 - \sigma(b+w^T x_i)) = \log(\sigma(-(b+w^T x_i)))
\]
  - For labels $y_i = +1/-1$ can write: $\sum_{i=1:n} \log(\sigma(y_i (b+w^T x_i)))$
Comparison with logistic regression

• In logistic regression (with labels +1/-1), the loss was:
  \[ \sum_{i=1:n} \log(\sigma(y_i(b+w^T x_i))) + \lambda \|w\|^2 \]
  – (we can include regularization)

• For support vector machines, we have the loss
  \[ \sum_{i=1:n} E^\infty(y_i(b+w^T x_i) - 1) + \lambda \|w\|^2 \]

• Where \(E^\infty(z)\) is 0 if \(z \geq 0\), or \(\infty\) otherwise. \(\lambda\) does not matter in this case!

• Only works for linearly separable data (for now!)
SVM formulation

- SVM problem: Min $\frac{1}{2} ||w||^2$
  w.r.t. $w$
  s.t. $y_i w^T x_i \geq 1$

- This can be solved with quadratic programming.
Non-linearly separable data

- A linear boundary might be too simple to capture the data.

- **Option 1:** *Relax the constraints* and allow some points to be misclassified by the margin.

- **Option 2:** *Allow a nonlinear decision boundary* in the input space by finding a linear decision boundary in an expanded space (*similar to adding polynomial terms in linear regression.*)
  - Here $x_i$ is replaced by $\phi(x_i)$, where $\phi$ is called a feature mapping.
Soften the primal objective

• We wanted to solve:
  \[ \min_w \quad \frac{1}{2} \|w\|^2 \]
  s.t. \[ y_i w^T x_i \geq 1 \]

• This can be re-written:
  \[ \min_w \quad \sum_i E_{\infty} (y_i (b + w^T x_i) - 1) + \frac{1}{2} \|w\|^2 \]
  where \[ \sum_i E_{\infty} (w^T x_i, y_i) = (\infty \text{ for a points inside margin, } 0 \text{ otherwise}) \]

• Soften misclassification cost:
  \[ \min_w \quad \sum_i L_{0-1} (w^T x_i, y_i) + \frac{1}{2} \|w\|^2 \]
  where \[ \sum_i L_{0-1} (w^T x_i, y_i) = (1 \text{ for a misclassification, } 0 \text{ correct classification}) \]

• But this is a non-convex objective!
Soften the primal objective

- **Non-convex objective?**

- Function is convex if: 
  average of \( f(x_1) \) and \( f(x_2) \) > \( f(x_1/2 + x_2/2) \) for all \( x_1, x_2 \).

- In a graph, this means: when you connect two lines on the graph, the connecting segment is never below the original line.

Zero-One loss

\[
z_i = y_i(b + w^T x_i)
\]
Soften the primal objective

• Non-convex objective?

• Function is convex if:
  average of \( f(x_1) \) and \( f(x_2) \) > \( f(x_1/2+x_2/2) \) for all \( x_1, x_2 \).

• In a graph, this means: when you connect two lines on the graph, the connecting segment is never below the original line.

Zero-One loss

\[ z_i = y_i(b + w^T x_i) \]
Approximation of the $L_{0-1}$ function

$$z_i = y_i(b + w^T x_i)$$
Approximation of the $L_{0-1}$ function

$$z_i = y_i(b + w^T x_i)$$

Hinge Loss

Copyright: C. Bishop, Pattern recognition and machine learning
Approximation of the $L_{0-1}$ function

Logistic regression

$z_i = y_i(b + w^T x_i)$

Copyright: C. Bishop, Pattern recognition and machine learning
Approximation of the $L_{0-1}$ function

$z_i = y_i (b + w^T x_i)$

Logistic regression

Quadratic loss

Hinge Loss

Copyright: C. Bishop, Pattern recognition and machine learning
SVM with hinge loss

• Hinge loss: \( L_{\text{hin}} (w^T x_i, y_i) = \max \{1 - y_i w^T x_i, 0\} \)

• Soften misclassification cost: 
  \[
  \min_w C \sum_i L_{\text{hin}} (w^T x_i, y_i) + \frac{1}{2} ||w||^2
  \]
  where \( C \) controls trade-off between slack penalty and margin.

• The hinge loss upper-bounds the 0-1 loss.
  \[
  L_{\text{hin}} (w^T x_i, y_i) \geq L_{0-1} (w^T x_i, y_i)
  \]
Primal Soft SVM problem

- Define slack variables \( \xi_i = L_{hin}(w^T x_i, y_i) = \max \{1-y_i (w^T x_i + b) 0\} \)

- Solve: \( \hat{w}_{soft} = \arg\min_{w, \xi} C \sum_{i=1:n} \xi_i + \frac{1}{2}||w||^2 \quad \text{Add Lagrange mult:} \)

s. t. \( y_i (w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n \quad \leq \alpha_i \)
\( \xi_i \geq 0, \quad i = 1, \ldots, n \quad \leq \beta_i \)

where \( w \in \mathbb{R}^m, \xi \in \mathbb{R}^n \)

- Introduce Lagrange multipliers: \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T, 0 \leq \alpha_i \)
\( \beta = (\beta_1, \beta_2, \ldots, \beta_n)^T, 0 \leq \beta_i \)
Soft SVM problem: Adding Lagrange multipliers

- **Primal** objective: \( (w, \xi, \alpha, \beta) = \arg \min_{w,\xi} \max_{\alpha,\beta} L(w, \xi, \alpha, \beta) \)

where \( L(w, \xi, \alpha, \beta) = \frac{1}{2}||w||^2 + C \sum_{i=1:n} \xi_i - \sum_{i=1:n} \alpha_i (y_i (w^T x_i + b) - 1 + \xi_i) - \sum_{i=1:n} \beta_i \xi_i \)
Soft SVM problem: Adding Lagrange multipliers

• **Primal** objective: \((w, \xi, \alpha, \beta) = \text{arg min}_{w,\xi} \text{max}_{\alpha,\beta} L(w, \xi, \alpha, \beta)\)

  where \(L(w, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C \sum_{i:1:n} \xi_i - \sum_{i:1:n} \alpha_i (y_i w^T x_i - 1 + \xi_i) - \sum_{i:1:n} \beta_i \xi_i\)

• **Dual** (invert min and max): \((w, \xi, \alpha, \beta) = \text{arg max}_{\alpha,\beta} \text{min}_{w,\xi} L(w, \xi, \alpha, \beta)\)
Soft SVM problem: Adding Lagrange multipliers

- **Primal** objective: \((w, \xi, \alpha, \beta) = \arg \min_{w,\xi} \max_{\alpha,\beta} L(w, \xi, \alpha, \beta)\)
  
  where \(L(w, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C \sum_{i=1:n} \xi_i - \sum_{i=1:n} \alpha_i (y_i w^T x_i - 1 + \xi_i) - \sum_{i=1:n} \beta_i \xi_i\)

- **Dual** (invert min and max): \((w, \xi, \alpha, \beta) = \arg \max_{\alpha,\beta} \min_{w,\xi} L(w, \xi, \alpha, \beta)\)

- Solve: \(\frac{\delta L}{\delta w} = w - \sum_i \alpha_i y_i x_i = 0 \Rightarrow w^* = \sum_i \alpha_i y_i x_i\)

  \(\frac{\delta L}{\delta \xi} = C1_n - \alpha - \beta = 0 \Rightarrow \beta = C1_n - \alpha\)

  Lagrange multipliers are positive, so we have: \(0 \leq \beta_i, 0 \leq \alpha_i \leq C\)
Soft SVM problem: Adding Lagrange multipliers

- **Primal** objective: \( (w, \xi, \alpha, \beta) = \arg\min_{w,\xi} \max_{\alpha,\beta} L(w, \xi, \alpha, \beta) \)

  where \( L(w, \xi, \alpha, \beta) = \frac{1}{2}||w||^2 + C \sum_{i=1:n} \xi_i - \sum_{i=1:n} \alpha_i (y_iw^T x_i - 1 + \xi_i) - \sum_{i=1:n} \beta_i \xi_i \)

- **Dual** (invert min and max): \( (w, \xi, \alpha, \beta) = \arg\max_{\alpha,\beta} \min_{w,\xi} L(w, \xi, \alpha, \beta) \)

- Solve:
  \[
  \frac{\delta L}{\delta w} = w - \sum_i \alpha_i y_i x_i = 0 \quad \Rightarrow \quad w^* = \sum_i \alpha_i y_i x_i
  \]
  \[
  \frac{\delta L}{\delta \xi} = C1_n - \alpha - \beta = 0 \quad \Rightarrow \quad \beta = C1_n - \alpha
  \]

  Lagrange multipliers are positive, so we have: \( 0 \leq \beta_i, 0 \leq \alpha_i \leq C \)

- Plug into dual:
  \[
  \max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)
  \]
  with constraints \( 0 \leq \alpha_i \leq C \) and \( \sum_i \alpha_i y_i = 0 \).

- This is a quadratic programming problem (similar to Hard SVM).
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C=\infty$, then Soft-SVM=>Hard-SVM.
**Soft SVM solution**

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C=\infty$, then Soft-SVM=>Hard-SVM.

- Points away from margin have $\alpha_i = 0, \xi_i=0$.
- The following have $\alpha_i > 0$
  - Points on the margin, $\xi_i=0$.
  - Points within the margin, $0 < \xi_i < 1$
  - Points on the decision line, $\xi_i = 1$.
  - Misclassified points, $\xi_i > 1$. 

\[
\mathbf{w}^T \mathbf{x} = 0
\]
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C=\infty$, then Soft-SVM=Hard-SVM.

- Points away from margin have $\alpha_i = 0, \xi_i=0$.
- The following have $\alpha_i > 0$
  - Points on the margin, $\xi_i=0$.
  - Points within the margin, $0 < \xi_i < 1$
  - Points on the decision line, $\xi_i = 1$.
  - Misclassified points, $\xi_i > 1$. 

\[
\alpha_j > 0, \xi_j = 0 \quad \alpha_j = 0 \quad w^T x = 0
\]
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C=\infty$, then Soft-SVM=>Hard-SVM.

- Points away from margin have $\alpha_i = 0$, $\xi_i=0$.
- The following have $\alpha_i > 0$
  - Points on the margin, $\xi_i=0$.
  - Points within the margin, $0 < \xi_i < 1$
  - Points on the decision line, $\xi_i = 1$.
  - Misclassified points, $\xi_i > 1$. 

$$w^T x = 0$$

$\alpha_j > 0$, $\xi_j = 0$

$0 < \xi_j < 1$
Soft SVM solution

• Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
• When $C \to \infty$, then Soft-SVM $\Rightarrow$ Hard-SVM.

• Points away from margin have $\alpha_i = 0$, $\xi_i = 0$.
• The following have $\alpha_i > 0$
  – Points on the margin, $\xi_i = 0$.
  – Points within the margin, $0 < \xi_i < 1$
  – Points on the decision line, $\xi_i = 1$.
  – Misclassified points, $\xi_i > 1$. 

\[ w^T x = 0 \]
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C = \infty$, then Soft-SVM $\rightarrow$ Hard-SVM.

- Points away from margin have $\alpha_i = 0$, $\xi_i = 0$.
- The following have $\alpha_i > 0$
  - Points on the margin, $\xi_i = 0$.
  - Points within the margin, $0 < \xi_i < 1$
  - Points on the decision line, $\xi_i = 1$.
  - Misclassified points, $\xi_i > 1$. 

\[ w^T x = 0 \]
Soft SVM solution

- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \rightarrow \infty$, then Soft-SVM=>Hard-SVM.
- Points away from margin have $\alpha_i = 0$, $\xi_i=0$.
- The following have $\alpha_i > 0$
  - Points on the margin, $\xi_i=0$.
  - Points within the margin, $0 < \xi_i < 1$
  - Points on the decision line, $\xi_i = 1$.
  - Misclassified points, $\xi_i > 1$.
- To predict on test data:
  $$h_w(x) = \text{sign}( \sum_{i=1:n} \alpha_i y_i (x_i \cdot x) )$$
- Only need to store the support vectors (i.e. points with $\alpha_i > 0$) to predict.
Soft SVM, Bias & Variance

- \( \hat{w}_{\text{soft}} = \text{argmin}_{w,\xi} \ C \sum_{i=1:n} \xi_i + \frac{1}{2}||w||^2 \)
- \( C \to \infty \), then Soft-SVM=>Hard-SVM

- What does lowering C do to bias and variance?
  - Increase bias and decrease variance?
  - Decrease bias and increase variance?
Non-linearly separable data

- A linear boundary might be too simple to capture the data.

- Option 1: Relax the constraints and allow some points to be misclassified by the margin.

- Option 2: Allow a nonlinear decision boundary in the input space by finding a linear decision boundary in an expanded space (similar to adding polynomial terms in linear regression.)
  - Here $x_i$ is replaced by $\phi(x_i)$, where $\phi$ is called a feature mapping.
Margin optimization in feature space

- Replacing $x_i$ by $\phi(x_i)$, the optimization problem for $w$ becomes:

  - **Primal form:**
    
    $\text{Min} \quad \frac{1}{2} ||w||^2$
    
    w.r.t. $w$
    
    s.t. $y_i w^T \phi(x_i) \geq 1$
Margin optimization in feature space

- Replacing \( x_i \) by \( \phi(x_i) \), the optimization problem for \( w \) becomes:

  - **Primal form:** \( \text{Min} \) \( \frac{1}{2} ||w||^2 \)
    
    w.r.t. \( w \)
    
    s.t. \( y_i(w^T \phi(x_i) + b) \geq 1 \)

  - **Dual form:** \( \text{Max} \) \( \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\phi(x_i) \cdot \phi(x_j)) \)
    
    w.r.t. \( \alpha_i \)
    
    s.t. \( \alpha_i \geq 0 \)
    
    \( \sum_i \alpha_i y_i = 0 \)
Feature space solution

• The optimal weights, in the expanded feature space, are
  \[ w = \sum_{i=1:n} \alpha_i y_i \phi(x_i) \]

• Classification of an input \( x \) is given by:
  \[ h_w(x) = \text{sign}( \sum_{i=1:n} \alpha_i y_i (\phi(x_i) \cdot \phi(x)) + b ) \]
Feature space solution

- The optimal weights, in the expended feature space, are
  \[ \mathbf{w} = \sum_{i=1:n} \alpha_i y_i \phi(x_i) \]
- Classification of an input \( \mathbf{x} \) is given by:
  \[ h_{\mathbf{w}}(\mathbf{x}) = \text{sign}(\sum_{i=1:n} \alpha_i y_i (\phi(x_i) \cdot \phi(x)) + b) \]
- Note that to solve the SVM optimization problem in dual form and to make a prediction, we only ever need to compute dot-products of feature vectors.
Feature space solution

- Potential problem:

- Let’s say we might need many feature – 3rd order?
- If we had m original features, now we have $1 + m + m^2 + m^3$
- If we had 10 original features, we now have more than 1000!
- In primal form this will be very costly to solve…
- Even in dual form we will need to compute the inner products!
Kernel functions

- Whenever a learning algorithm (such as SVMs) can be written in terms of dot-products, it can be generalized to kernels.

- A kernel is any function $K: \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$, which corresponds to a dot product for some feature mapping $\phi$:

  $$K(x_1, x_2) = \phi(x_1) \cdot \phi(x_2) \text{ for some } \phi$$
Kernel functions

- Whenever a learning algorithm (such as SVMs) can be written in terms of dot-products, it can be generalized to kernels.

- A kernel is any function $K: \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$, which corresponds to a dot product for some feature mapping $\phi$:

$$K(x_1, x_2) = \phi(x_1) \cdot \phi(x_2)$$

- Conversely, by choosing feature mapping $\phi$, we implicitly choose a kernel function.

- Recall that $\phi(x_1) \cdot \phi(x_2) = |\phi(x_1)| \cdot |\phi(x_2)| \cdot \cos \angle(\phi(x_1), \phi(x_2))$, where $\angle$ denotes the angle between the vectors, so a kernel function can be thought of as a notion of similarity.
Example: Quadratic kernel

• Let $K(x, z) = (x \cdot z)^2$.

• Is this a kernel?

$$K(x, z) = (\sum_{i=1}^{m} x_i z_i)(\sum_{j=1}^{m} x_j z_j)$$
$$= \sum_{i,j \in \{1..m\}} x_i z_i x_j z_j$$
$$= \sum_{i,j \in \{1..m\}} (x_i x_j)(z_i z_j)$$
Example: Quadratic kernel

- Let $K(x, z) = (x \cdot z)^2$.

- Is this a kernel?

\[
K(x, z) = (\sum_{i=1}^{m} x_i z_i) (\sum_{j=1}^{m} x_j z_j)
\]
\[
= \sum_{i,j \in \{1..m\}} x_i z_i x_j z_j
\]
\[
= \sum_{i,j \in \{1..m\}} (x_i x_j) (z_i z_j)
\]

- We see it is a kernel, with feature mapping:

\[
\phi(x) = < x_1^2, x_1 x_2, \ldots, x_1 x_m, x_2 x_1, x_2^2, \ldots, x_m^2>
\]

Feature vector includes all squares of elements and all cross terms.
Example: Quadratic kernel

- Let \( K(x, z) = (x \cdot z)^2 \).

- Is this a kernel?

\[
K(x, z) = \left( \sum_{i=1:m} x_i z_i \right) \left( \sum_{j=1:m} x_j z_j \right) \\
= \sum_{i,j \in \{1..m\}} x_i z_i x_j z_j \\
= \sum_{i,j \in \{1..m\}} (x_i x_j) (z_i z_j)
\]

- We see it is a kernel, with feature mapping:

\[
\phi(x) = < x_1^2, x_1 x_2, \ldots, x_1 x_m, x_2 x_1, x_2^2, \ldots, x_m^2 >
\]

Feature vector includes all squares of elements and all cross terms.

**Important:** Computing \( \phi \) takes \( O(m^2) \) but computing \( K \) only takes \( O(m) \).
Polynomial kernels

- More generally, \( K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^d \) is a kernel, for any positive integer \( d \):
  \[
  K(\mathbf{x}, \mathbf{z}) = (\sum_{i=1:m} x_i z_i)^d
  \]
- If we expanded the sum above in the naïve way, we get \( m^d \) terms.
- Terms are monomials (products of \( x_i \)) with total power equal to \( d \).
Polynomial kernels

- More generally, \( K(x, z) = (x \cdot z)^d \) is a kernel, for any positive integer \( d \):
  \[
  K(x, z) = (\sum_{i=1:m} x_i z_i)^d
  \]
- If we expanded the sum above in the naïve way, we get \( m^d \) terms.
- Terms are monomials (products of \( x_i \)) with total power equal to \( d \).
- If we use the primal form of the SVM, each term gets a weight.
- **Curse of dimensionality**: it is very expensive both to optimize and to predict with an SVM in primal form.
- However, evaluating the dot-product of any two feature vectors can be done using \( K \) in \( O(m) \).
The “kernel trick”

• If we work with the dual, we do not have to ever compute the feature mapping $\phi$. We just compute the similarity kernel $K$. 
The “kernel trick”

• If we work with the dual, we do not have to ever compute the feature mapping $\phi$. We just compute the similarity kernel $K$.

• We just replace any inner product $\phi(x_1) \cdot \phi(x_2)$ with $K(x_1, x_2)$

• This is justified as any valid kernel defines an inner product with some feature expansion – even if we do not want to actually form this feature expansion (sometimes it is even impossible)
The “kernel trick”

• We can solve the dual for the $\alpha_i$ with the kernel trick

(just replace $\phi(x_1) \cdot \phi(x_2)$ with $K(x_1, x_2)$)

$$\begin{align*}
\text{Max} & \quad \sum_{i=1:n} \alpha_i - \frac{1}{2} \sum_{i,j=1:n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\
\text{w.r.t.} & \quad \alpha_i \\
\text{s.t.} & \quad \alpha_i \geq 0 \quad \text{and} \quad \sum_{i:1..n} \alpha_i y_i = 0
\end{align*}$$

• The class of a new input $x$ is computed as:

$$h_w(x) = \text{sign}(\sum_{i=1:n} \alpha_i y_i K(x_i, x))$$

where $x_i$ are the support vectors (defining the margin).

• Remember, $K(\cdot, \cdot)$ can be evaluated in $O(m)$ time = big savings!
Some other kernel functions

- \( K(x, z) = (1 + x \cdot z)^d \) - feature expansion has all monomial terms of total power.

- Radial basis / Gaussian kernel: \( K(x, z) = \exp(-||x-z||^2 / 2\sigma^2) \)
  - This kernel has an infinite-dimensional feature expansion, but dot-products can still be computed in \( O(m) \) (where \( m=\#\)original features)
  - It would be impossible to use this representation in a ‘primal’ formulation!

- Sigmoidal kernel: \( K(x, z) = \tanh(c_1 x \cdot z + c_2) \)
Example: Gaussian kernel

Note the non-linear decision boundary
Non-parametric

- Remember that we call models that cannot be expressed with a fixed (finite) number of parameters non-parametric.
- With Gaussian kernels our feature expansion is infinite-dimensional.
- Thus, SVM with a Gaussian kernel is another example of a non-parametric model.
- Same advantages and disadvantages we discussed in Lecture 7 (Instance-based learning).
Kernels beyond SVMs

- A lot of research related to defining kernel functions suitable to particular tasks / kinds of inputs (e.g. words, graphs, images).

- Many kernels are available:
  - Information diffusion kernels (Lafferty and Lebanon, 2002)
  - Diffusion kernels on graphs (Kondor and Jebara, 2003)
  - String kernels for text classification (Lodhi et al, 2002)
  - String kernels for protein classification (Leslie et al, 2002)
  ... and others!
Example: String kernels

- Very important for DNA matching, text classification, …

- Often use a sliding window of length $k$ over the two strings that we want to compare.

- Within the fixed-size window we can do many things:
  - Count exact matches.
  - Weigh mismatches based on how bad they are.
  - Count certain markers, e.g. AGT.

- The kernel is the sum of these similarities over the two sequences.
Kernelizing other ML algorithms

• Many other machine learning algorithms have a “dual formulation”, in which dot-products of features can be replaced by kernels.

• Examples:
  – Perceptron
  – Logistic regression
  – Linear regression (We’ll do this later in the course!)
Multiple classes

• One-vs-All: Learn K separate binary classifiers.
  – Can lead to inconsistent results.
  – Training sets are imbalanced, e.g. assuming n examples per class, each binary classifier is trained with positive class having 1*n of the data, and negative class having (K-1)*n of the data.
Multiple classes

• One-vs-All: Learn K separate binary classifiers.
  – Can lead to inconsistent results.
  – Training sets are imbalanced, e.g. assuming n examples per class, each binary classifier is trained with positive class having 1*n of the data, and negative class having (K-1)*n of the data.

• Multi-class SVM: Define the margin to be the gap between the correct class and the nearest other class.
SVMs for regression

• Minimize a regularized error function:

\[ \hat{w} = \text{argmin}_w \ C \sum_{i=1:n} (y_i - w^T x_i)^2 + \frac{1}{2}||w||^2 \]

• Introduce slack variables to optimize “tube” around the regression function.
SVMs for regression

- Minimize a regularized error function:
  \[ \hat{w} = \arg\min_w \ C \sum_{i=1:n} (y_i - w^T x_i)^2 + \frac{1}{2}\|w\|^2 \]

- Introduce slack variables to optimize “tube” around the regression function.

- Typically, relax to \( \varepsilon \)-sensitive error on the linear target to ensure sparse solution (i.e. few support vectors):
  \[ \hat{w} = \arg\min_w \ C \sum_{i=1:n} E_\varepsilon (y_i - w^T x_i)^2 + \frac{1}{2}\|w\|^2 \]
  where \( E_\varepsilon = \begin{cases} 0 & \text{if } (y_i - w^T x_i) < \varepsilon, \\ (y_i - w^T x_i) - \varepsilon & \text{otherwise} \end{cases} \]
What you should know

From last class and from today:

- Perceptron algorithm.
- Margin definition for linear SVMs.
- Use of Lagrange multipliers to transform optimization problems.
- Primal and dual optimization problems for SVMs.
- Feature space version of SVMs.
- The kernel trick and examples of common kernels.