COMP 551 – Applied Machine Learning
Lecture 12: Support Vector Machines (cont'd)

Instructor: Joelle Pineau (*jpineau@cs.mcgill.ca)*

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

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

Today's quiz

Apply Support Vector Machines to data generated by the AND boolean function:

Y = X1 AND X2

X1	X2	Y
0	0	-1
0	1	-1
1	0	-1
1	1	+1

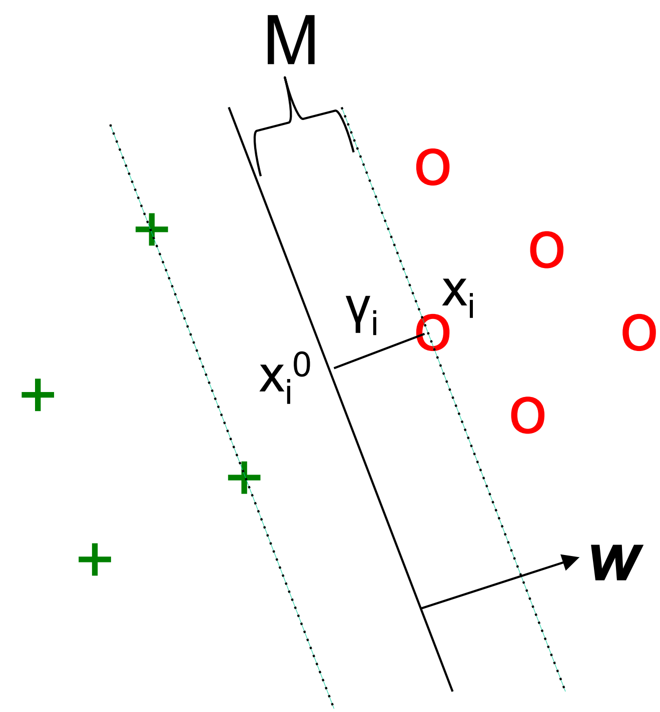
1. What is *M*, the margin size?
2. What are the weights, *w**?
3. Which datapoints define the margin?

SVM formulation

- SVM problem: Min $\frac{1}{2}\,||\mathbf{w}||^2$

  w.r.t. w

  s.t. $y_i \mathbf{w}^T \mathbf{x}_i \geq 1$



- 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 ϕ is called a feature mapping.

Soften the primal objective

- We wanted to solve: $\min_{\mathbf{w}} \quad \frac{1}{2}\,||\mathbf{w}||^2$
  s.t. $\quad y_i\mathbf{w}^T\mathbf{x}_i \geq 1$

- This can be re-written: $\min_{\mathbf{w}} \quad \sum_i L_{0\text{-}\infty}(w^Tx_i,\ y_i) + \frac{1}{2}\,||\mathbf{w}||^2$
  s.t. $\quad y_i\mathbf{w}^T\mathbf{x}_i \geq 1$

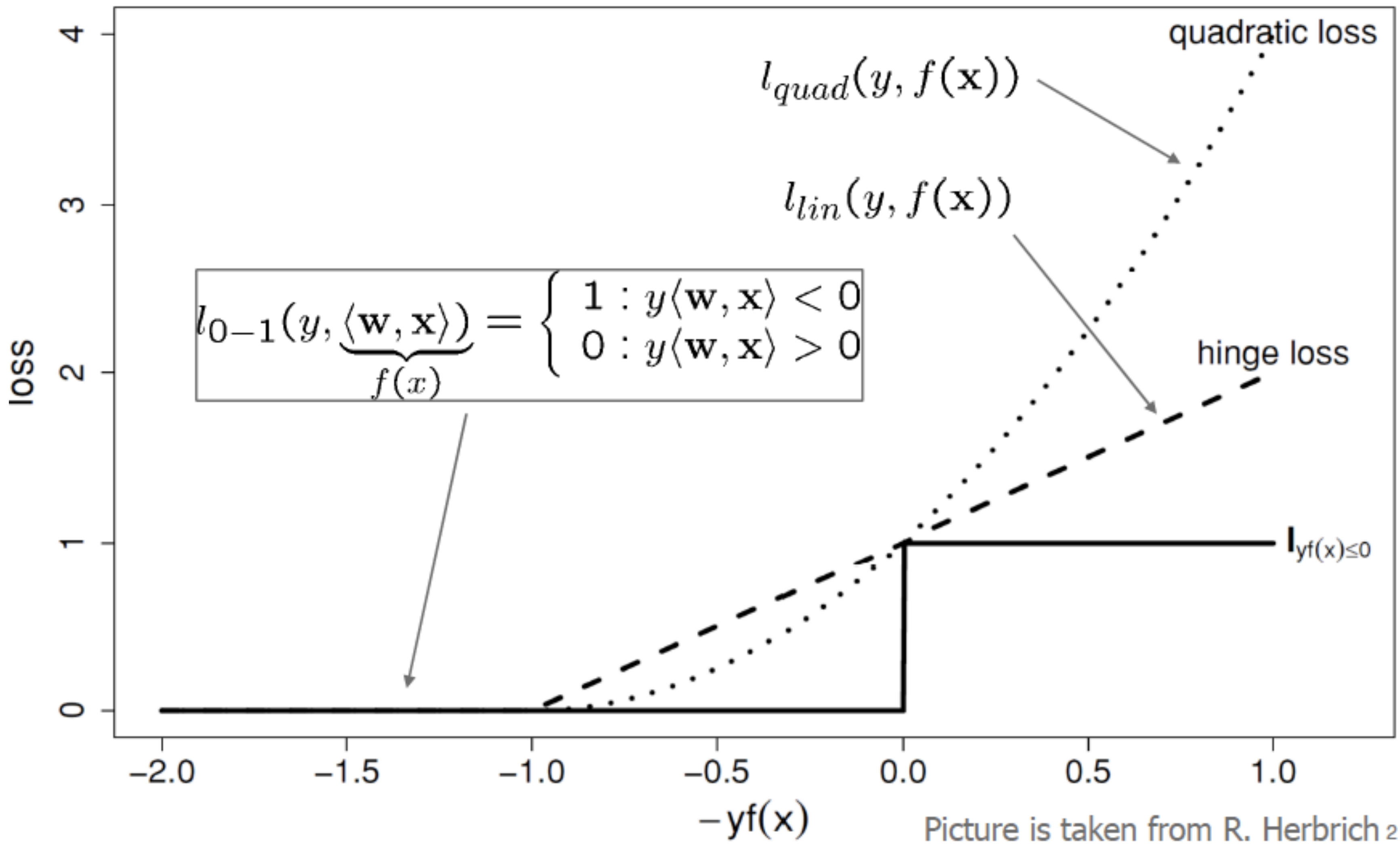
  where $\sum_i L_{0\text{-}\infty}(w^Tx_i,\ y_i) = (\infty$ for a misclassification, 0 correct classification$)$

- Soften misclassification cost: $\min_{\mathbf{w}} \quad \sum_i L_{0\text{-}1}(w^Tx_i,\ y_i) + \frac{1}{2}\,||\mathbf{w}||^2$
  s.t. $\quad y_i\mathbf{w}^T\mathbf{x}_i \geq 1$

  where $\sum_i L_{0\text{-}1}(w^Tx_i,\ y_i) = (1$ for a misclassification, 0 correct classification$)$

- But this is a **non-convex** objective!

Approximation of the $L_{0\text{-}1}$ function



Picture is taken from R. Herbrich 2

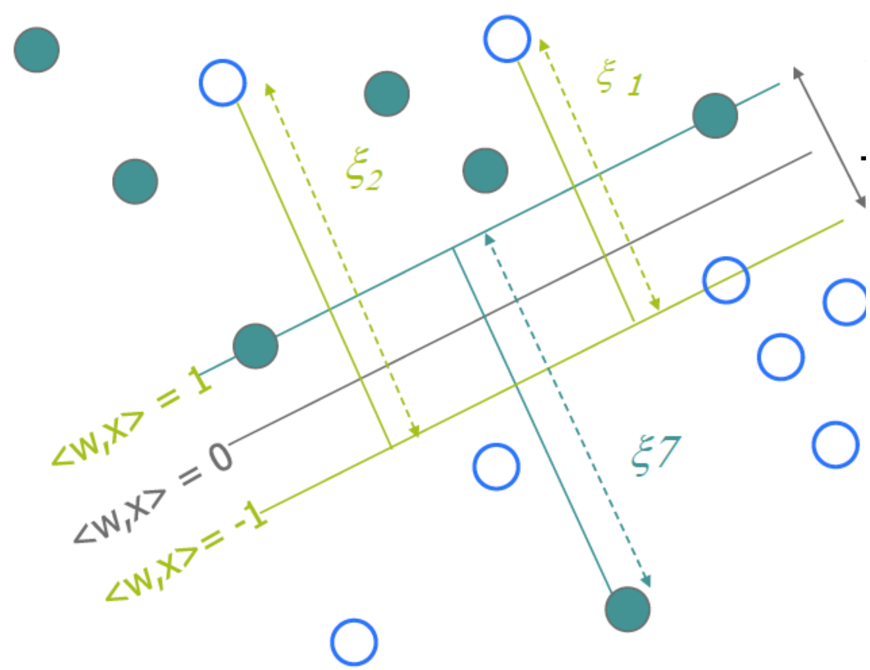
SVM with hinge loss

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

- Soften misclassification cost: $min_{\mathbf{w}}\ C\ \sum_i L_{hin}\,(w^T x_i,\, y_i) + ½\ ||\mathbf{w}||^2$
  where C controls trade-off between slack penalty and margin.

- The hinge loss upper-bounds the 0-1 loss.

$\xi_i\ \geq\ 1\text{-}\ y_i \mathbf{w}^T \mathbf{x}_i \geq L_{0\text{-}1}\,(w^T x_i,\, y_i)$



Primal Soft SVM problem

- Define slack variables $\xi_i = L_{hin}(w^T x_i, y_i) = max\ \{1\text{-}y_i \mathbf{w}^T \mathbf{x}_i , 0\}$

- Solve: $\hat{\mathbf{w}}_{soft} = argmin_{\mathbf{w},\xi}\ C \sum_{i:1:n} \xi_i + ½||\mathbf{w}||^2$ Add Lagrange mult:

  s. t. $y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i ,\quad i = 1, \ldots, n$ <= Call this α_i

  $\xi_i \geq 0 ,\quad i = 1, \ldots, n$ <= Call this β_i

  where $w\ \varepsilon\ R^m,\ \xi\ \varepsilon\ R^n$

- Introduce Lagrange multipliers: $\boldsymbol{\alpha} = (\ \alpha_1, \alpha_2, \ldots, \alpha_n\ )^T,\ 0 \leq \alpha_i$

  $\boldsymbol{\beta} = (\ \beta_1, \beta_2, \ldots, \boldsymbol{\alpha}_n\ )^T,\ 0 \leq \beta_i$

Soft SVM problem: Adding Lagrange multipliers

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

where $L(w, \xi, \alpha, \beta) = ½||w||^2 + C \sum_{i:1:n} \xi_i - \sum_{i:1:n} \alpha_i (y_i \mathbf{w}^T \mathbf{x}_i - 1 + \xi_i) - \sum_{i:1:n} \beta_i \xi_i$

Soft SVM problem: Adding Lagrange multipliers

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

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

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

Soft SVM problem: Adding Lagrange multipliers

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

  where $L(w, \xi, \alpha, \beta) = ½||w||^2 + C \sum_{i:1:n} \xi_i - \sum_{i:1:n} \alpha_i (y_i \mathbf{w}^T \mathbf{x}_i - 1 + \xi_i) - \sum_{i:1:n} \beta_i \xi_i$

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

- Solve: $\delta L/\delta \mathbf{w} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$

  $\delta L/\delta \boldsymbol{\xi} = C1_n - \boldsymbol{\alpha} - \boldsymbol{\beta} = 0 \quad \Rightarrow \quad \boldsymbol{\beta} = C1_n - \boldsymbol{\alpha}$

  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: $(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\min_{\boldsymbol{w},\boldsymbol{\xi}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

  where $L(w, \xi, \alpha, \beta) = ½||w||^2 + C \sum_{i:1:n} \xi_i - \sum_{i:1:n} \alpha_i (y_i \boldsymbol{w}^T \boldsymbol{x}_i - 1 + \xi_i) - \sum_{i:1:n} \beta_i \xi_i$

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

- Solve: $\delta L/\delta \boldsymbol{w} = \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x}_i = 0 \quad \Rightarrow \quad \boldsymbol{w}^* = \sum_i \alpha_i y_i \boldsymbol{x}_i$

  $\delta L/\delta \boldsymbol{\xi} = C1_n - \boldsymbol{\alpha} - \boldsymbol{\beta} = 0 \quad \Rightarrow \quad \boldsymbol{\beta} = C1_n - \boldsymbol{\alpha}$

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

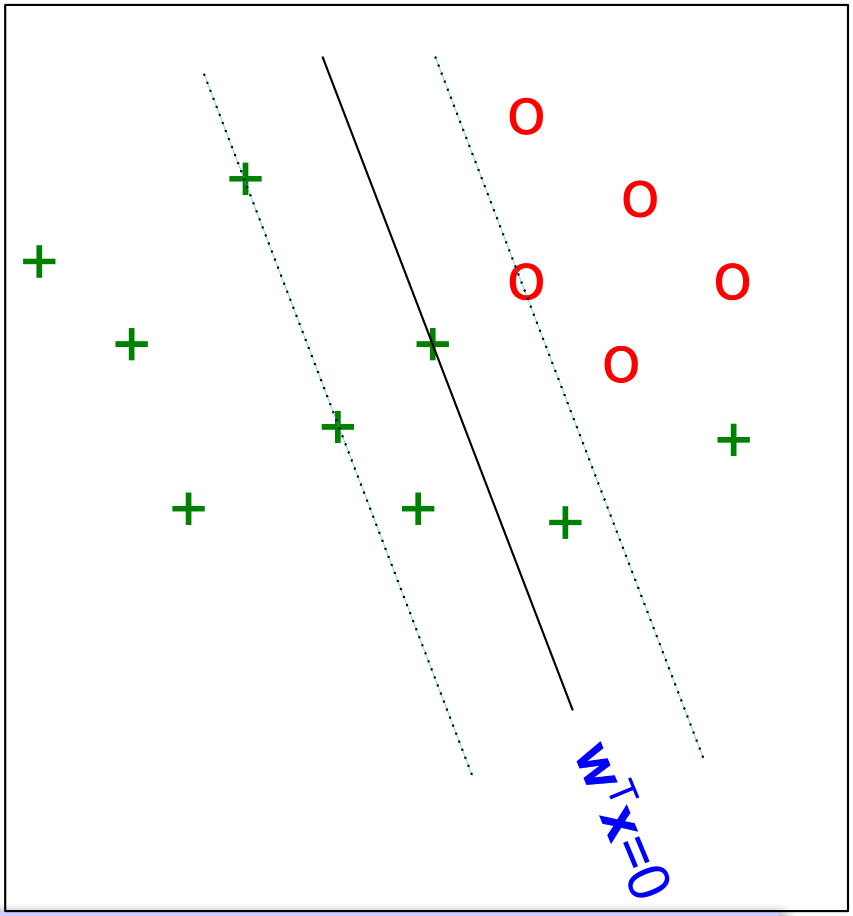
- Plug into dual : $\max_\alpha \sum_i \alpha_i - ½ \sum_{i,j} y_i y_j \alpha_i \alpha_j (\boldsymbol{x}_i \cdot \boldsymbol{x})$

  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

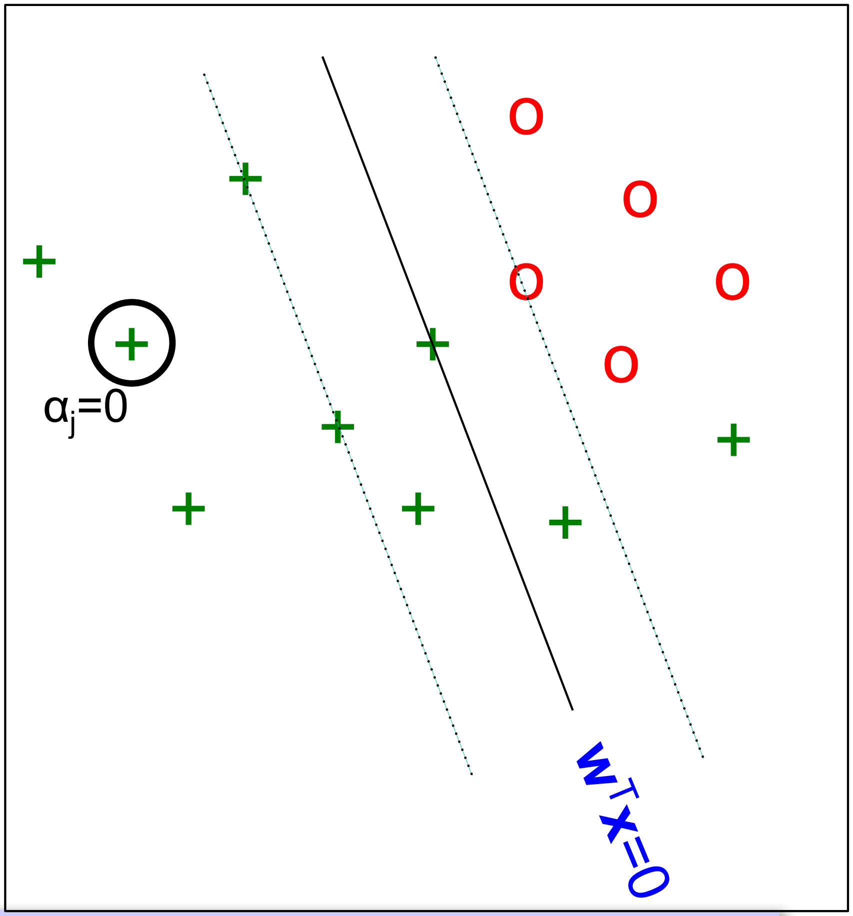
- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \Rightarrow \infty$, then Soft-SVM=>Hard-SVM.



Soft SVM solution

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

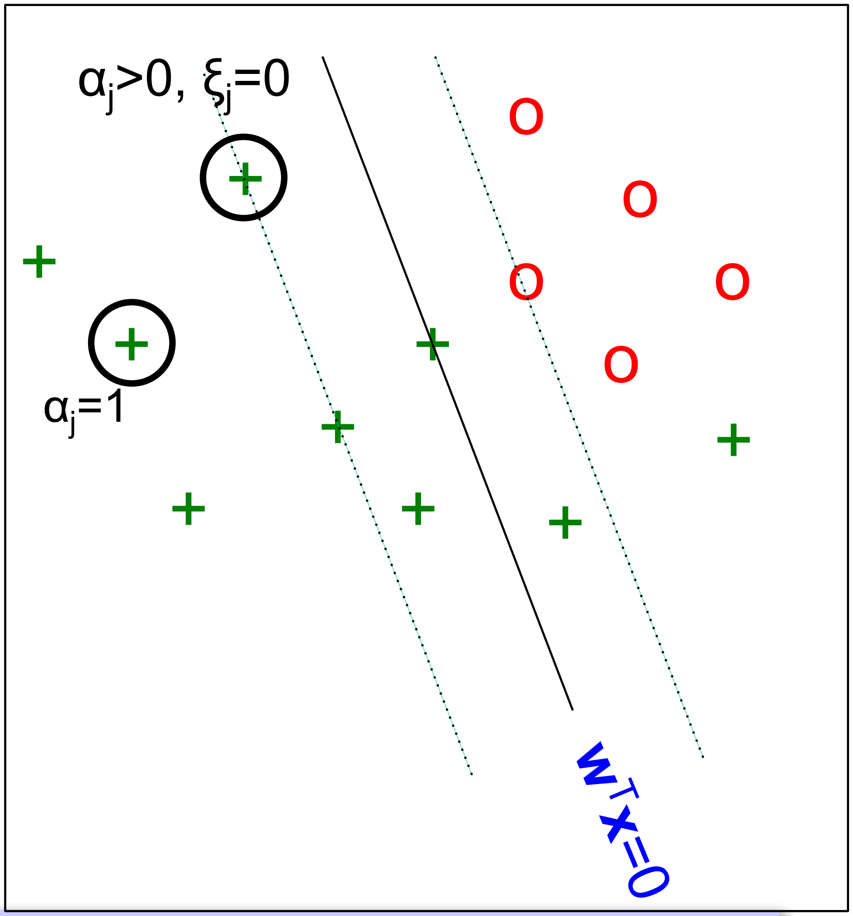
- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i = 0$.
- Points within the margin have $0 < \xi_i < 1$
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.



Soft SVM solution

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

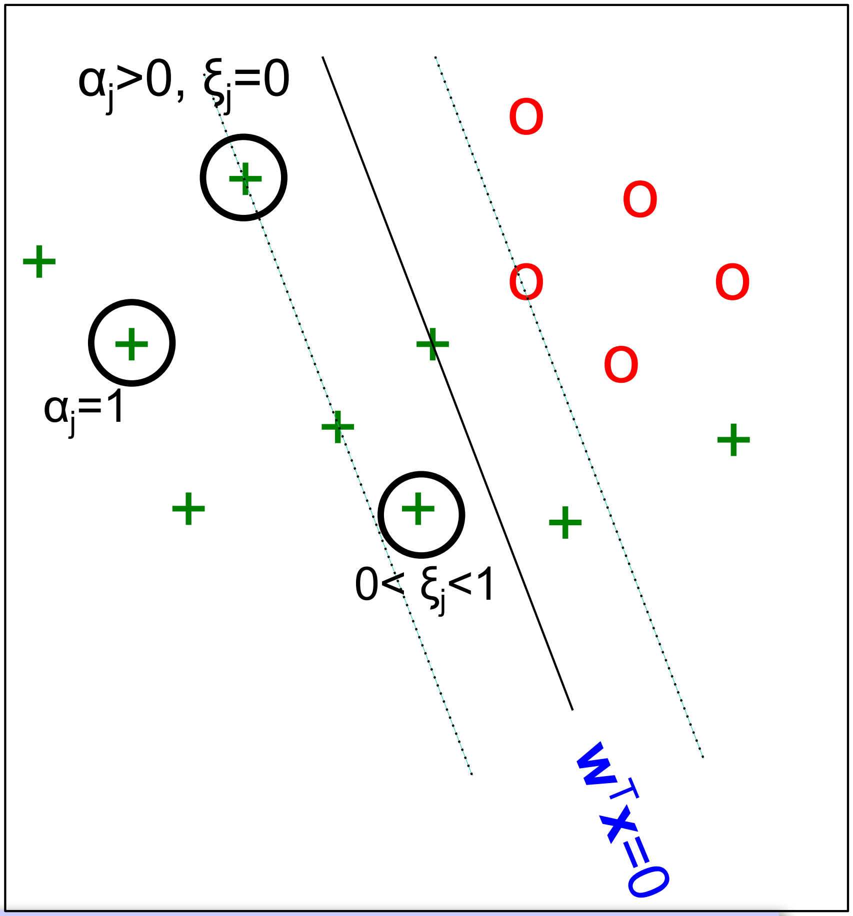
- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i=0$.
- Points within the margin have $0 < \xi_i < 1$
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.



Soft SVM solution

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

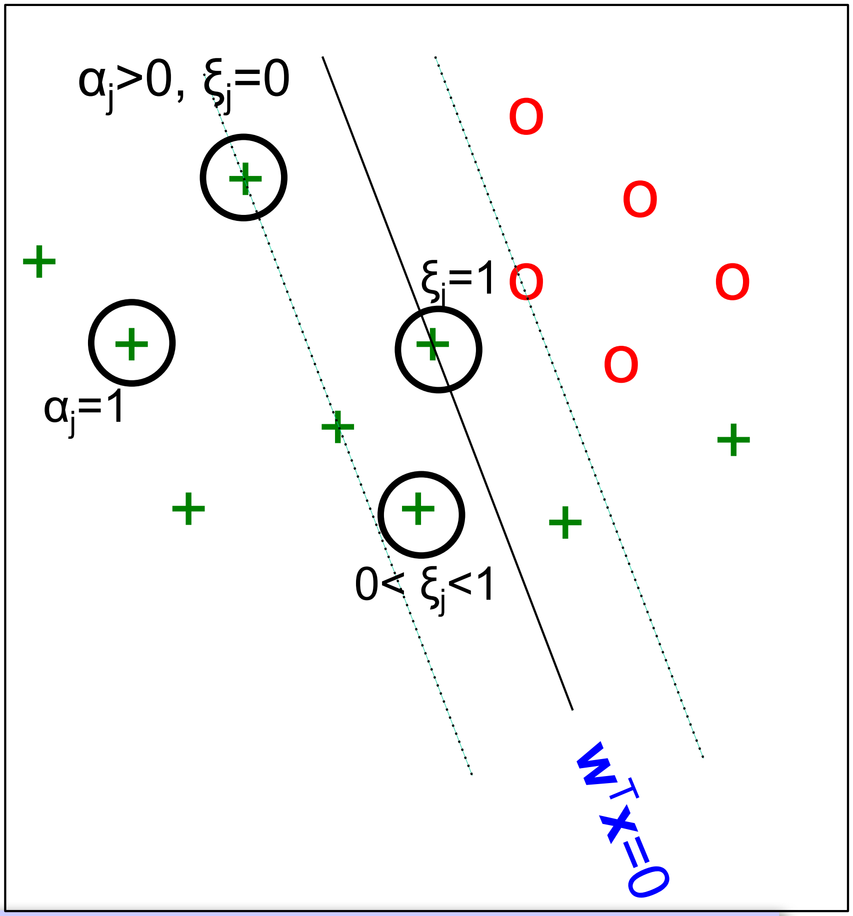
- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i=0$.
- Points within the margin have $0 < \xi_i < 1$
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.



Soft SVM solution

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

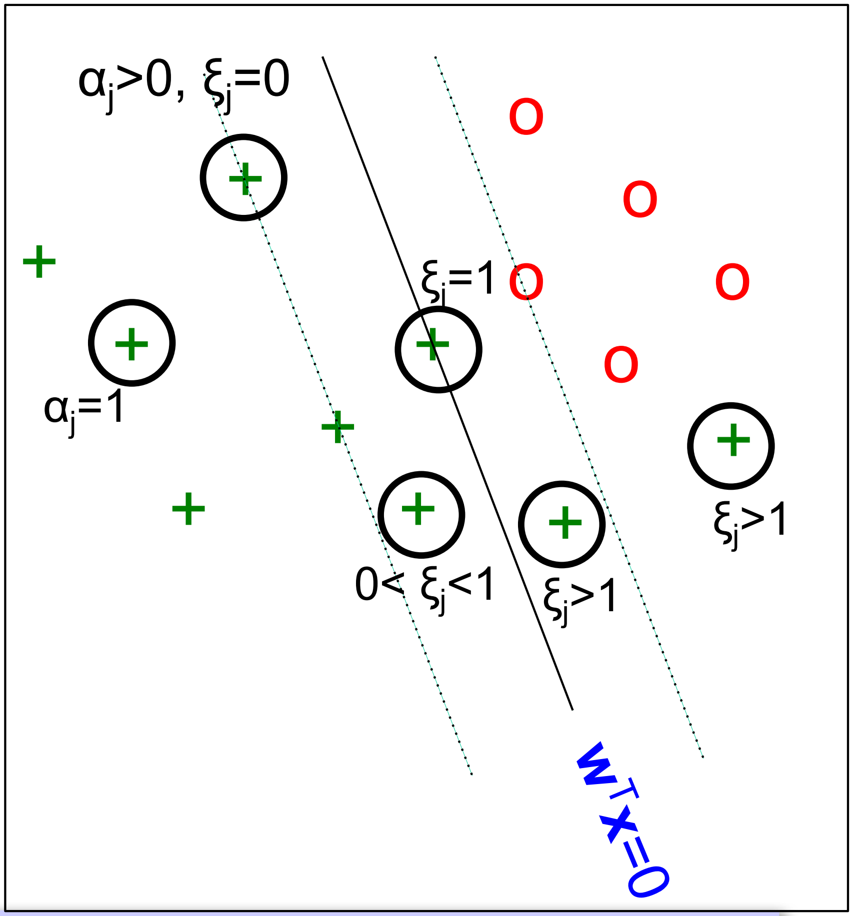
- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i=0$.
- Points within the margin have $0 < \xi_i < 1$
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.



Soft SVM solution

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

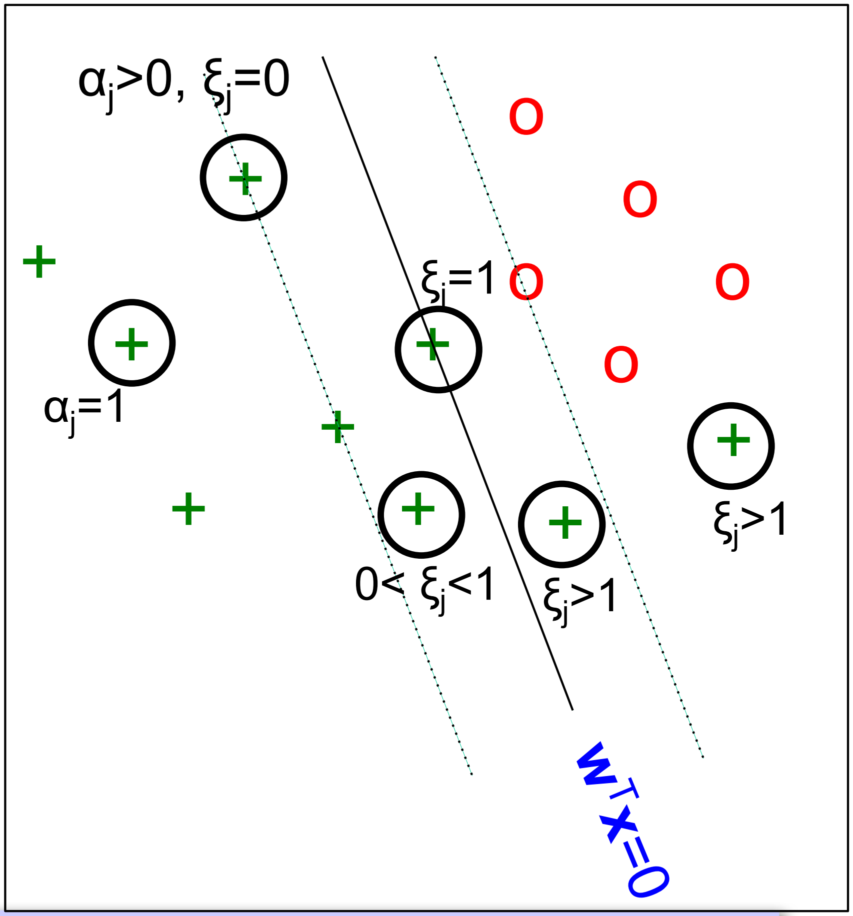
- Points away from margin have $\alpha_i = 0$.
- Points on the margin have $\alpha_i > 0$ and $\xi_i=0$.
- Points within the margin have $0 < \xi_i < 1$
- Points on the decision line have $\xi_i = 1$.
- Misclassified points have $\xi_i > 1$.



Soft SVM solution

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

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



- To predict on test data:

$$h_w(\mathbf{x}) = sign(\sum_{i=1:n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}))$$

- Only need to store the support vectors (i.e. points on the margin) to predict.

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.

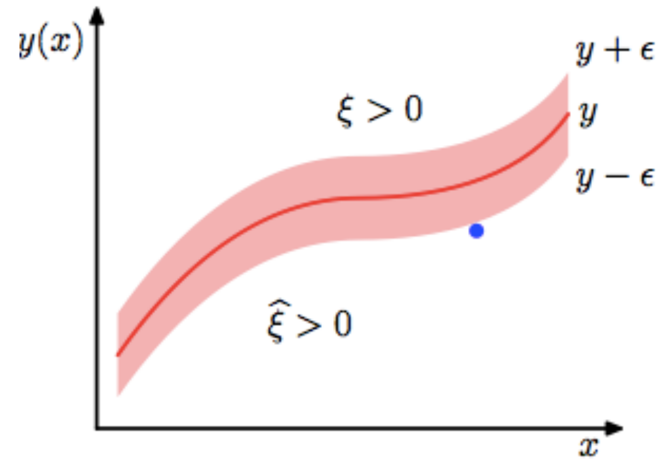
- Multi-class SVM: Define the margin to be the gap between the correct class and the nearest other class.

SVMs for regression

- Minimize a regularized error function:

$$\hat{\boldsymbol{w}} = \mathrm{argmin}_{\boldsymbol{w}}\ C \sum_{i:1:n} (y_i - w^T x_i)^2 + \tfrac{1}{2}||\boldsymbol{w}||^2$$

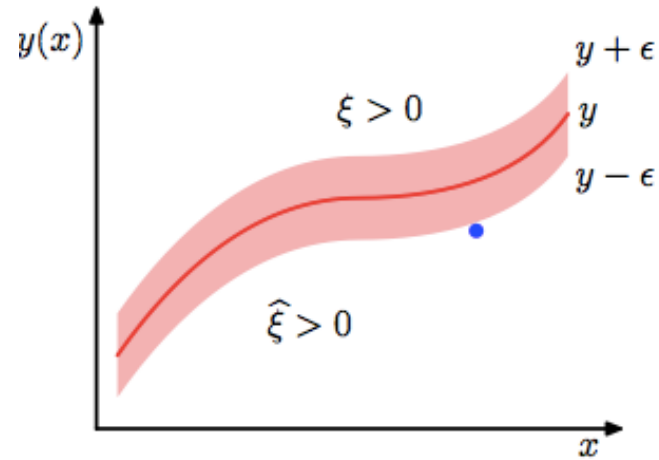
- Introduce slack variables to optimize "tube" around the regression function.



SVMs for regression

- Minimize a regularized error function:

$$\hat{\mathbf{w}} = \mathrm{argmin}_{\mathbf{w}} \; C \sum_{i:1:n} (y_i - w^T x_i)^2 + ½||\mathbf{w}||^2$$



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

- Typically, relax to ε-sensitive error on the linear target to ensure sparse solution (i.e. few support vectors):

$$\hat{\mathbf{w}} = \mathrm{argmin}_{\mathbf{w}} \; C \sum_{i:1:n} E_\varepsilon (y_i - w^T x_i)^2 + ½||\mathbf{w}||^2$$

where $E_\varepsilon = \begin{cases} 0 & \text{if } (y_i - w^T x_i) < \varepsilon, \\ (y_i - w^T x_i) - \varepsilon & \text{otherwise} \end{cases}$

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 ϕ is called a feature mapping.

Margin optimization in feature space

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

  – Primal form: $\quad Min \quad \frac{1}{2}||\mathbf{w}||^2$

  $\quad$ w.r.t. $\quad \mathbf{w}$

  $\quad$ s.t. $\quad y_i \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1$

Margin optimization in feature space

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

  – Primal form: *Min* $\frac{1}{2}||\mathbf{w}||^2$
  w.r.t. $\mathbf{w}$
  s.t. $y_i \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1$

  – Dual form: *Max* $\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}))$
  w.r.t. α_i
  s.t. $\alpha_i \geq 0$
  $\sum_i \alpha_i y_i = 0$

Feature space solution

- The optimal weights, in the expended feature space, are

$$\boldsymbol{w} = \textstyle\sum_{i=1:n} \alpha_i y_i \phi(\boldsymbol{x}_i)$$

- Classification of an input $\boldsymbol{x}$ is given by:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = sign(\textstyle\sum_{i=1:n} \alpha_i y_i (\phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x})))$$

Feature space solution

- The optimal weights, in the expended feature space, are

$$\boldsymbol{w} = \sum_{i=1:n} \alpha_i y_i \phi(\boldsymbol{x}_i)$$

- Classification of an input $\boldsymbol{x}$ is given by:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = sign(\sum_{i=1:n} \alpha_i y_i (\phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x})))$$

- 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.

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 $K: R^m \times R^m \rightarrow R$, which corresponds to a dot product for some feature mapping ϕ:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2) \text{ for some } \phi$$

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 $K: R^m \times R^m \rightarrow R$, which corresponds to a dot product for some feature mapping ϕ:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2) \text{ for some } \phi$$

- Conversely, by choosing feature mapping ϕ, we implicitly choose a kernel function.

- Recall that $\phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2) = \cos\angle(\mathbf{x}_1, \mathbf{x}_2)$, 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 $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^2$.
- Is this a kernel?

$$
\begin{aligned}
K(\mathbf{x}, \mathbf{z}) &= \left(\textstyle\sum_{i=1:m} x_i z_i \right) \left(\textstyle\sum_{j=1:m} x_j z_j \right) \\
&= \textstyle\sum_{i,j \varepsilon \{1..m\}} x_i z_i x_j z_j \\
&= \textstyle\sum_{i,j \varepsilon \{1..m\}} (x_i x_j) (z_i z_j)
\end{aligned}
$$

Example: Quadratic kernel

- Let $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^2$.
- Is this a kernel?

$$
\begin{aligned}
K(\mathbf{x}, \mathbf{z}) &= \left(\textstyle\sum_{i=1:m} x_i z_i \right) \left(\textstyle\sum_{j=1:m} x_j z_j \right) \\
&= \textstyle\sum_{i,j \varepsilon \{1..m\}} x_i z_i x_j z_j \\
&= \textstyle\sum_{i,j \,\varepsilon\, \{1..m\}} (x_i x_j) (z_i z_j)
\end{aligned}
$$

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

$$\phi(\mathbf{x}) = < x_1^2, x_1 x_2, \ldots, x_1 x_m, x_2 x_1, x_2^2, \ldots, x_m^2 >$$

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

Example: Quadratic kernel

- Let $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^2$.
- Is this a kernel?

$$
\begin{aligned}
K(\mathbf{x}, \mathbf{z}) &= \left(\textstyle\sum_{i=1:m} x_i z_i \right) \left(\textstyle\sum_{j=1:m} x_j z_j \right) \\
&= \textstyle\sum_{i,j \varepsilon \{1..m\}} x_i z_i x_j z_j \\
&= \textstyle\sum_{i,j \varepsilon \{1..m\}} (x_i x_j) (z_i z_j)
\end{aligned}
$$

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

$$\phi(\mathbf{x}) = < x_1^2, x_1 x_2, \ldots, x_1 x_m, x_2 x_1, x_2^2, \ldots, x_m^2 >$$

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

Important: Computing ϕ 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 n^d terms.
- Terms are monomials (products of x_i) with total power equal to d.

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 n^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-produce 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 ϕ. 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 ϕ. We just compute the similarity kernel K.

- We can solve the dual for the α_i :

Max	$\sum_{i=1:n} \alpha_i - ½ \sum_{i,j=1:n} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$
w.r.t.	α_i
s.t.	$\alpha_i \geq 0$ and $\sum_{i:1..n} \alpha_i y_i = 0$

The "kernel trick"

- If we work with the dual, we do not have to ever compute the feature mapping ϕ. We just compute the similarity kernel K.

- We can solve the dual for the α_i :

  Max $\quad \sum_{i=1:n} \alpha_i - \frac{1}{2} \sum_{i,j=1:n} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$

  w.r.t. $\quad \alpha_i$

  s.t. $\quad \alpha_i \geq 0$ and $\sum_{i:1..n} \alpha_i y_i = 0$

- The class of a new input x is computed as:

  $h_{\mathbf{w}}(\mathbf{x}) = sign(\sum_{i=1:n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}))$

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

- Remember, $K(\cdot, \cdot)$ can be evaluated in $O(m)$ time = big savings!

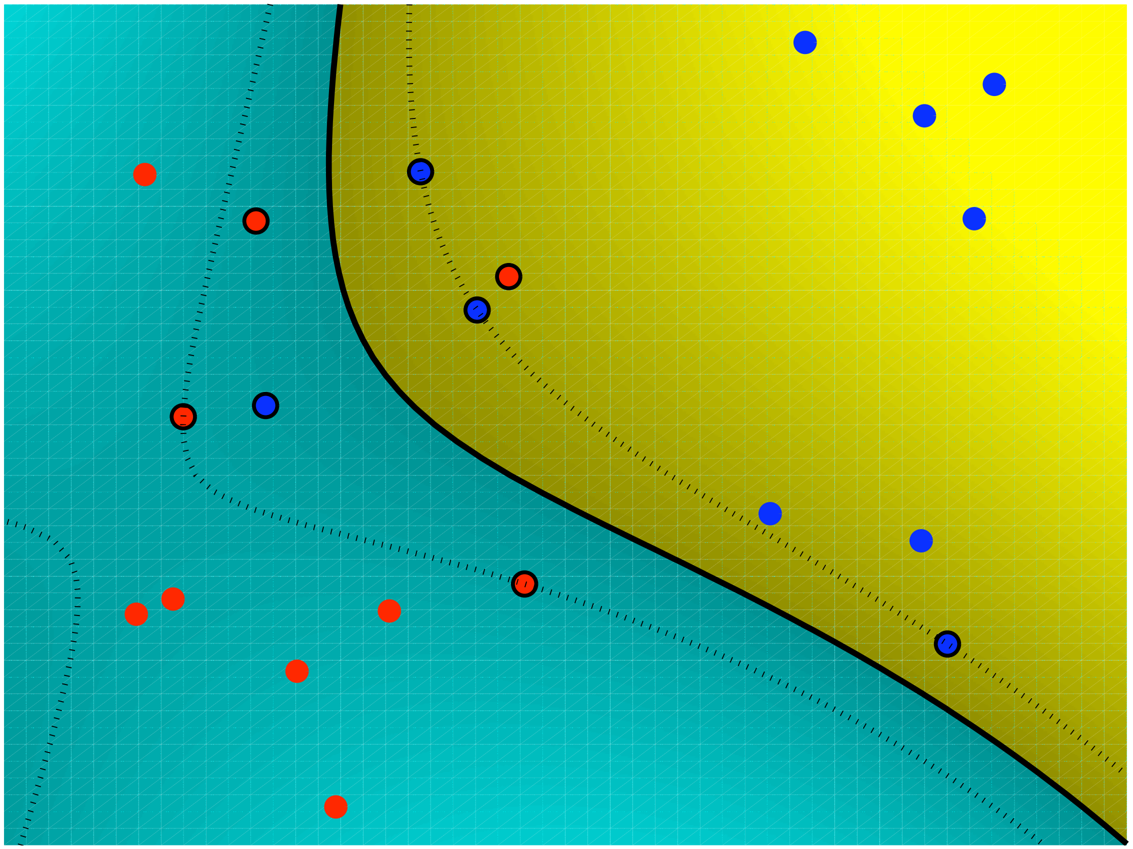
Some other kernel functions

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

- Radial basis / Gaussian kernel: $K(\mathbf{x}, \mathbf{z}) = exp\ (-||\mathbf{x}-\mathbf{z}||^2 / 2\sigma^2)$
  - This kernel has an infinite-dimensional feature expansion, but dot-products can still be computed in $O(m)$ (where m=#features)

- Sigmoidal kernel: $K(\mathbf{x}, \mathbf{z}) = tanh(c_1 \mathbf{x} \cdot \mathbf{z} + c_2)$

Example: Gaussian kernel



Note the non-linear decision boundary

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

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.