COMP 551 – Applied Machine Learning
Lecture 9: Support Vector Machines (cont'd)

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

Slides mostly by: Joelle Pineau

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

Quizzes

Quizzes for:

- Instance based learning
- Support vector machines

Now available for self-test

Recap primal and dual

- Consider both solutions:

$$p^* = \min_{\mathbf{w}} \max_{\alpha:\alpha_i \geq 0} L(\mathbf{w},\alpha) \qquad \textbf{Primal}$$

$$d^* = \max_{\alpha:\alpha_i \geq 0} \min_{\mathbf{w}} L(\mathbf{w},\alpha) \qquad \textbf{Dual}$$

- The optimal $\mathbf{w}^*$, $\boldsymbol{\alpha}^*$ 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

- $d^* = \max_{\alpha:\alpha i \geq 0} \min_{\mathbf{w}} L(\mathbf{w},\alpha)$ **Dual**

- **Solution strategy: First step:** solve the inner problem ($\min_{\mathbf{w}} L(\mathbf{w},\alpha)$) :
- Taking derivatives of $L(\mathbf{w}, \boldsymbol{\alpha})$ wrt $\mathbf{w}$, setting to 0, and solving for $\mathbf{w}$:

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \tfrac{1}{2} ||\mathbf{w}||^2 + \sum_i \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i))$$

$$\delta L/\delta \mathbf{w} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0$$

$$\mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$$

- Just like for the perceptron with zero initial weights, the optimal solution $\mathbf{w}^*$ is a linear combination of the $\mathbf{x}_i$.
- We can plug this in, and now find the $\boldsymbol{\alpha}$ that maximize the outer problem

Recap primal and dual

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

$$d^* = \max_{\alpha:\alpha_i \geq 0} \min_{\mathbf{w}} L(\mathbf{w},\alpha)$$

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \tfrac{1}{2} \|\mathbf{w}\|^2 + \sum_i \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i))$$

$$\mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$$

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

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

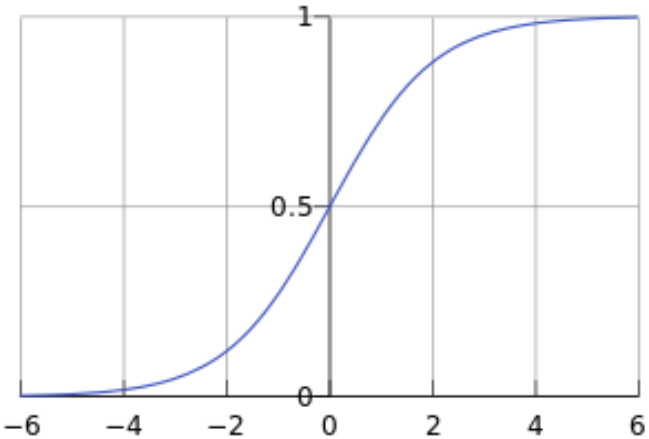
$$\max_{\alpha:\alpha_i \geq 0} \tfrac{1}{2} \| \mathbf{w}^* \|^2 + \sum_i \alpha_i (1 - y_i (\mathbf{w}^{*T} \mathbf{x}_i))$$, *use a bit of algebra to get*

$$\max_{\alpha:\alpha_i \geq 0} \sum_i \alpha_i - \tfrac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^T \mathbf{x})$$

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:
  - $\log(\sigma(b+\mathbf{w}^T\mathbf{x}_i))$ *for samples from the positive class*
  - $\log(1-\sigma(b+\mathbf{w}^T\mathbf{x}_i))$ *for samples from the negative class*



$\log(1-\sigma(b+\mathbf{w}^T\mathbf{x}_i)) = \log(\sigma(-(b+\mathbf{w}^T\mathbf{x}_i)))$

  - *For labels* $y_i = +1/-1$ *can write:* $\sum_{i=1:n} \log(\sigma(y_i\,(b+\mathbf{w}^T\mathbf{x}_i)))$

Comparison with logistic regression

- In logistic regression (with labels +1/-1), the loss was:

  $\sum_{i=1:n} log(\sigma(y_i(b+ \mathbf{w}^T\mathbf{x}_i))) + \lambda \|\mathbf{w}\|^2$

  – (we can include regularization)

- For support vector machines, we have the loss

  $\sum_{i=1:n} E_\infty (y_i(b+w^Tx_i)-1) + \lambda \|\mathbf{w}\|^2$

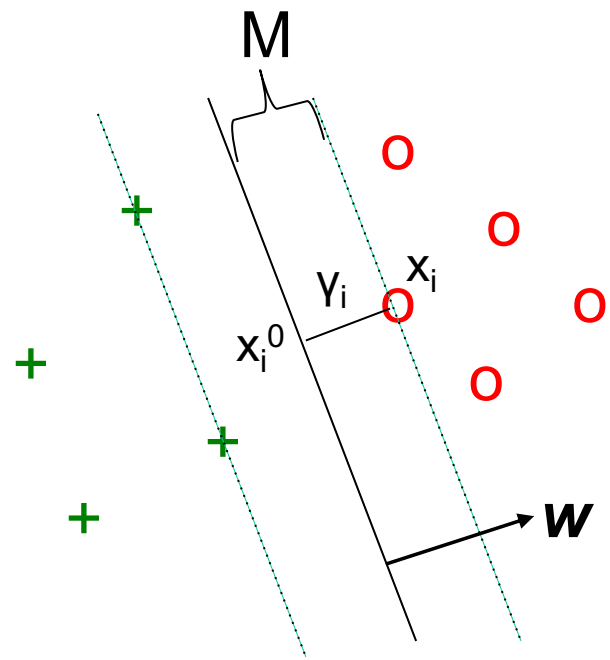
- Where $E_\infty(z)$ is 0 if $z \geq 0$, or ∞ otherwise. λ does not matter in this case!

- Only works for linearly separable data (for now!)

SVM formulation

- SVM problem:

$$\text{Min} \quad \tfrac{1}{2}\,||\boldsymbol{w}||^2$$
$$\text{w.r.t.} \quad \boldsymbol{w}$$
$$\text{s.t.} \quad y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 1$$



- 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: $\min_{\mathbf{w}} \quad \sum_i E_\infty (y_i(b+w^T x_i)-1) \; + \; \frac{1}{2} \|\mathbf{w}\|^2$
  where $\sum_i E_\infty (w^T x_i, y_i) = (\infty$ for a points inside margin, 0 otherwise$)$

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

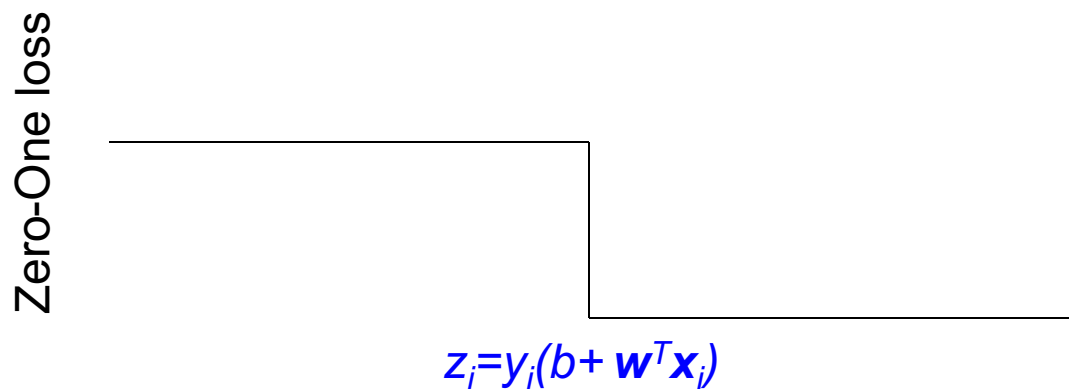
- But this is a **non-convex** objective!

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

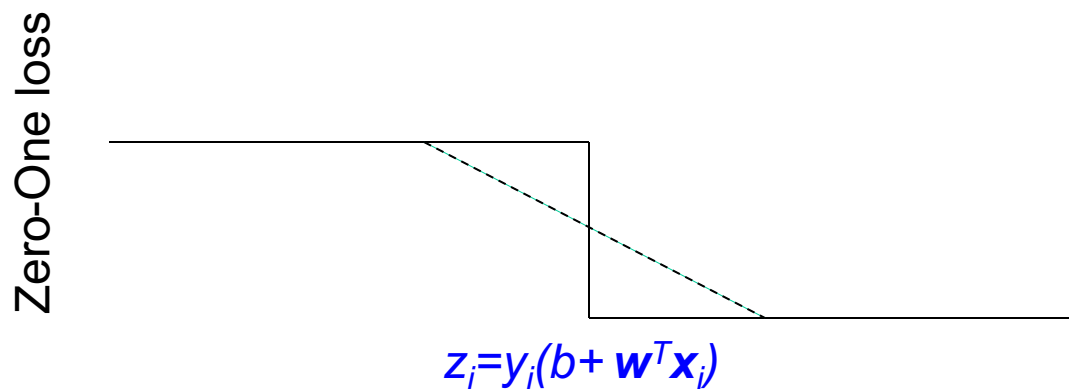


Soften the primal objective

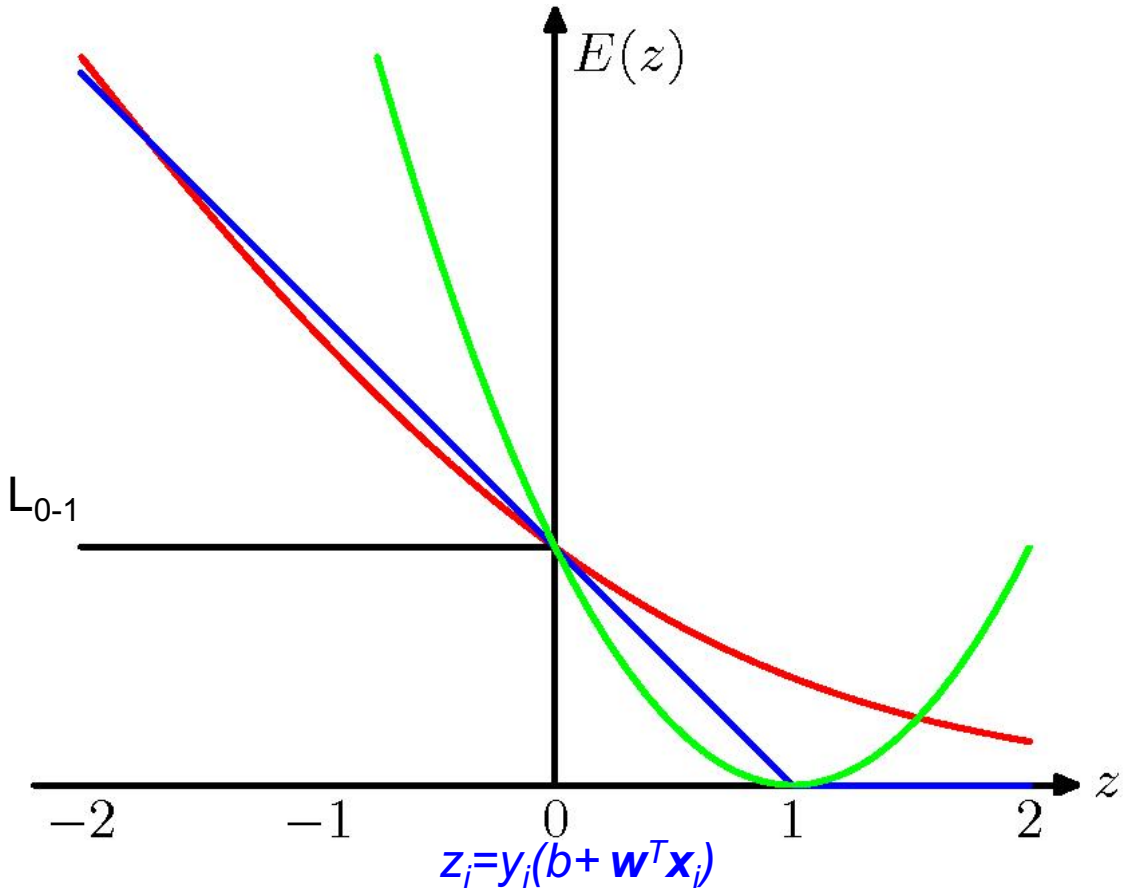
- Non-convex objective?

- 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

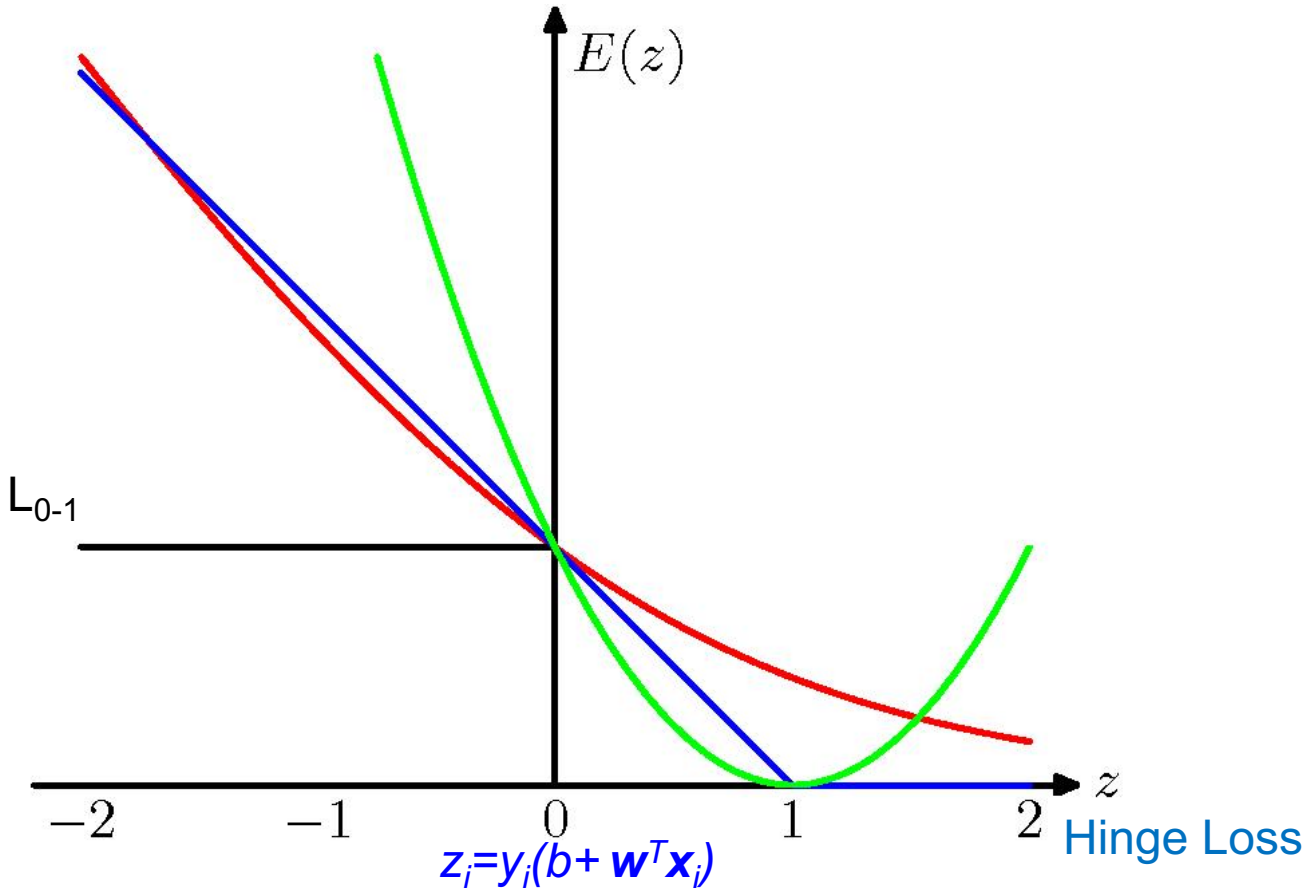


Approximation of the $L_{0\text{-}1}$ function



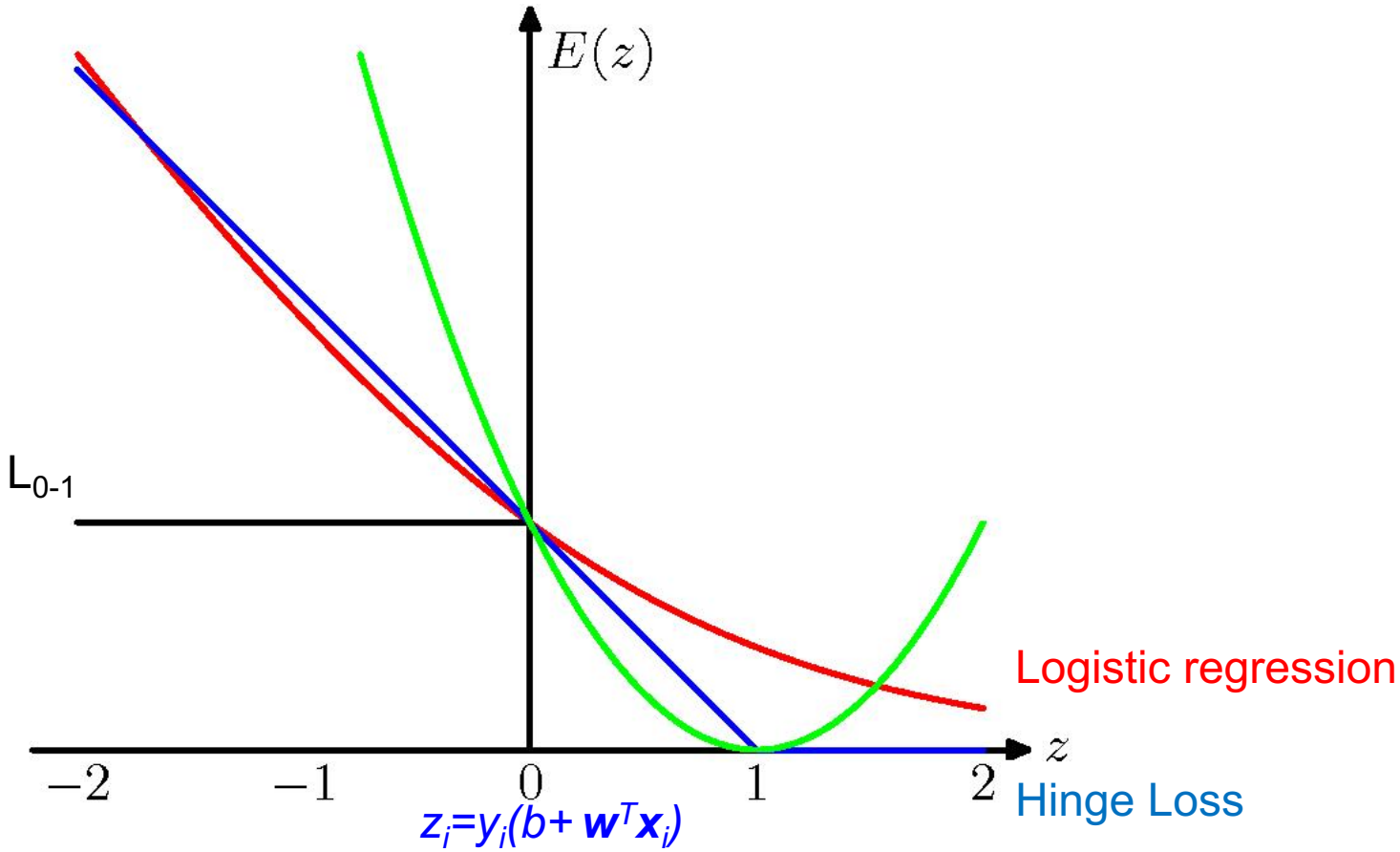
Copyright: C. Bishop, Pattern recognition and machine learning

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



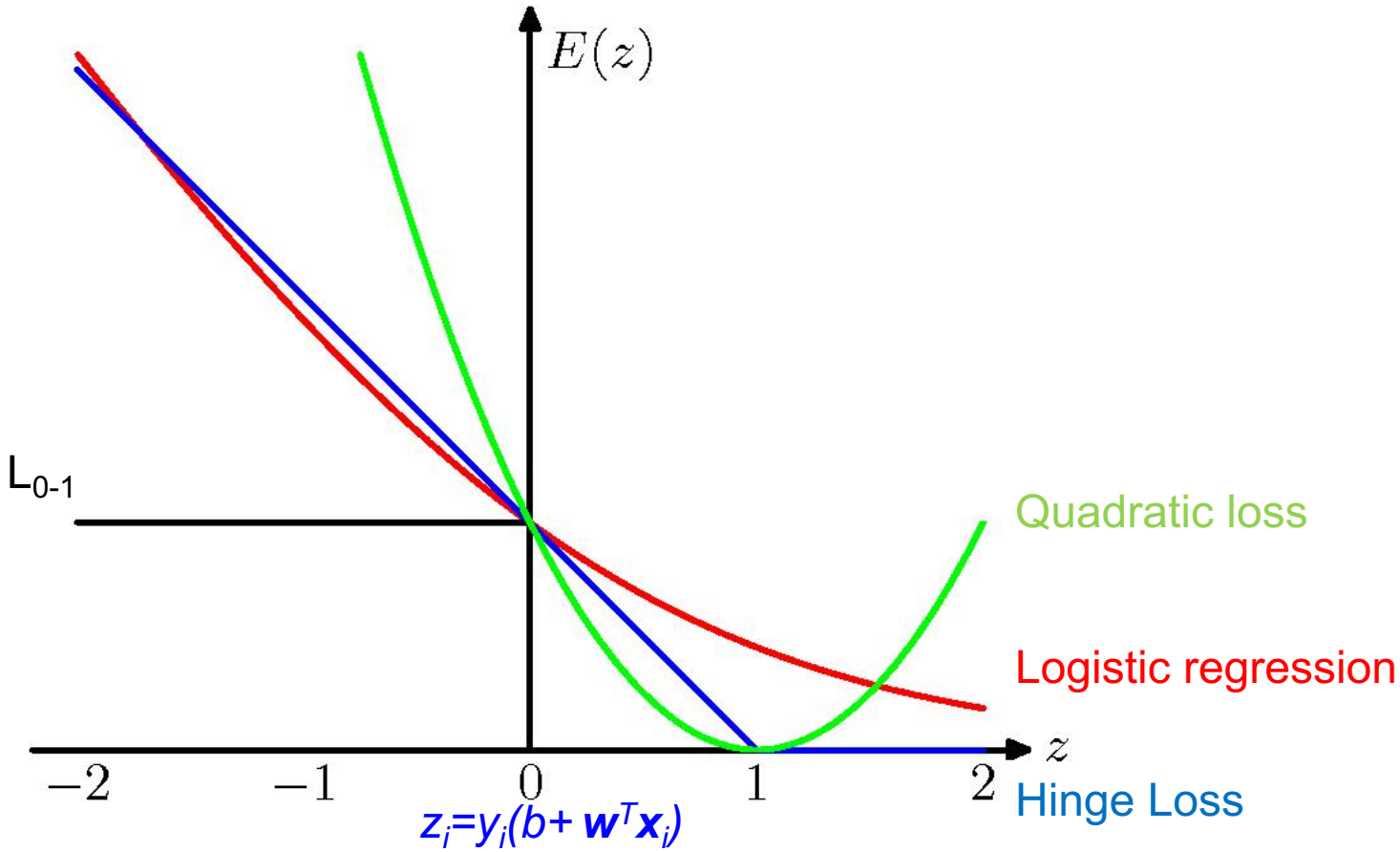
Copyright: C. Bishop, Pattern recognition and machine learning

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



Copyright: C. Bishop, Pattern recognition and machine learning

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



Copyright: C. Bishop, Pattern recognition and machine learning

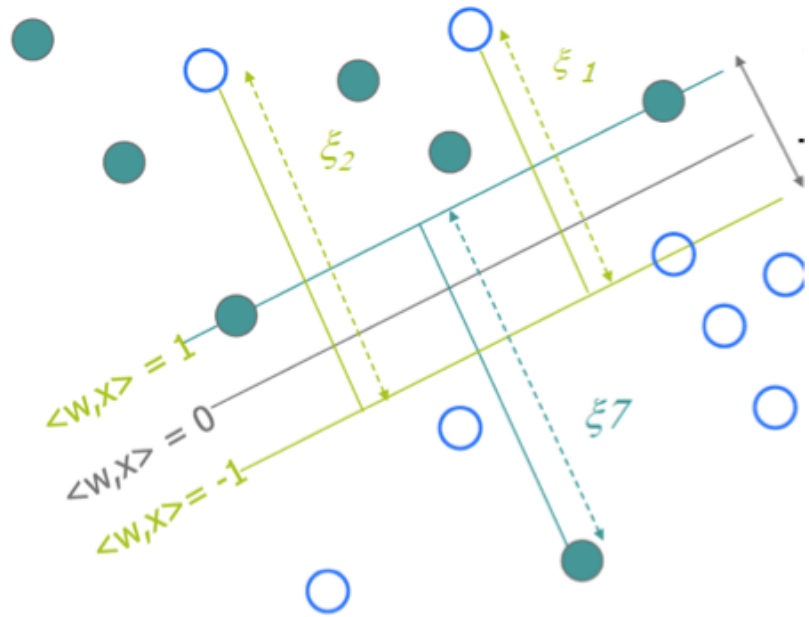
SVM with hinge loss

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

- 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.

$L_{hin}(w^T x_i, y_i) \geq L_{0\text{-}1}(w^T x_i, y_i)$



Primal Soft SVM problem

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

- 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. $y_i(\mathbf{w}^T\mathbf{x}_i+b) \geq 1 - \xi_i,\quad i = 1, \ldots, n \quad <= \alpha_i$

  $\xi_i \geq 0,\quad i = 1, \ldots, n \quad <= \beta_i$

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

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

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

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

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

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

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: $0 \le \beta_i$, $0 \le \alpha_i \le C$

Soft SVM problem: Adding Lagrange multipliers

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

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

- 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: $0 \le \beta_i$, $0 \le \alpha_i \le C$

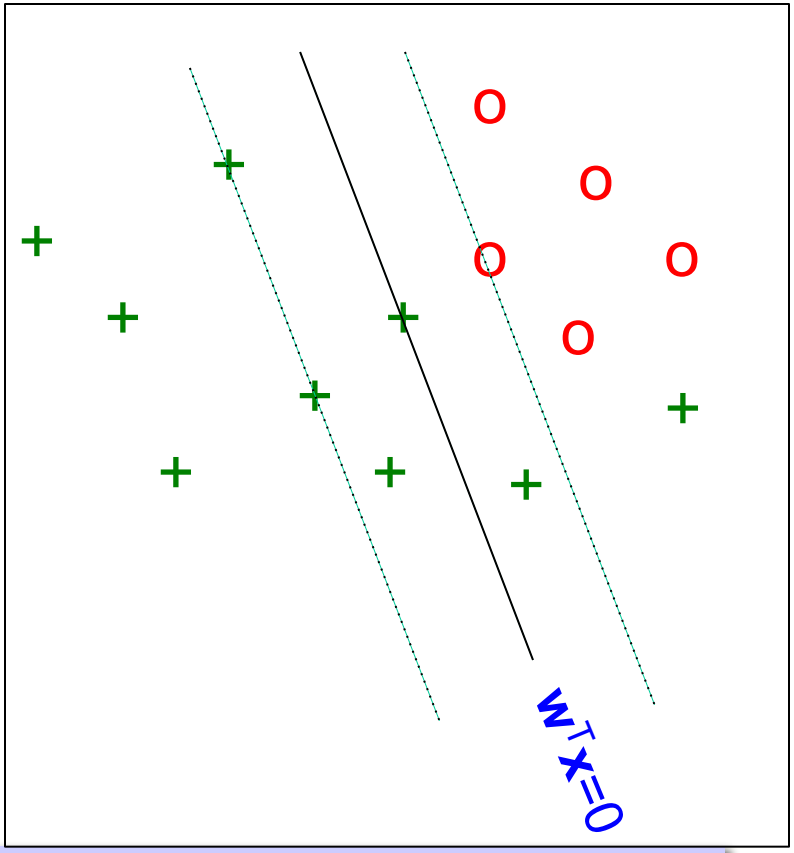
- Plug into dual : $\max_\alpha \sum_i \alpha_i - ½ \sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x})$

  with constraints $0 \le \alpha_i \le C$ and $\sum_i \alpha_i y_i = 0$.

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

Soft SVM solution

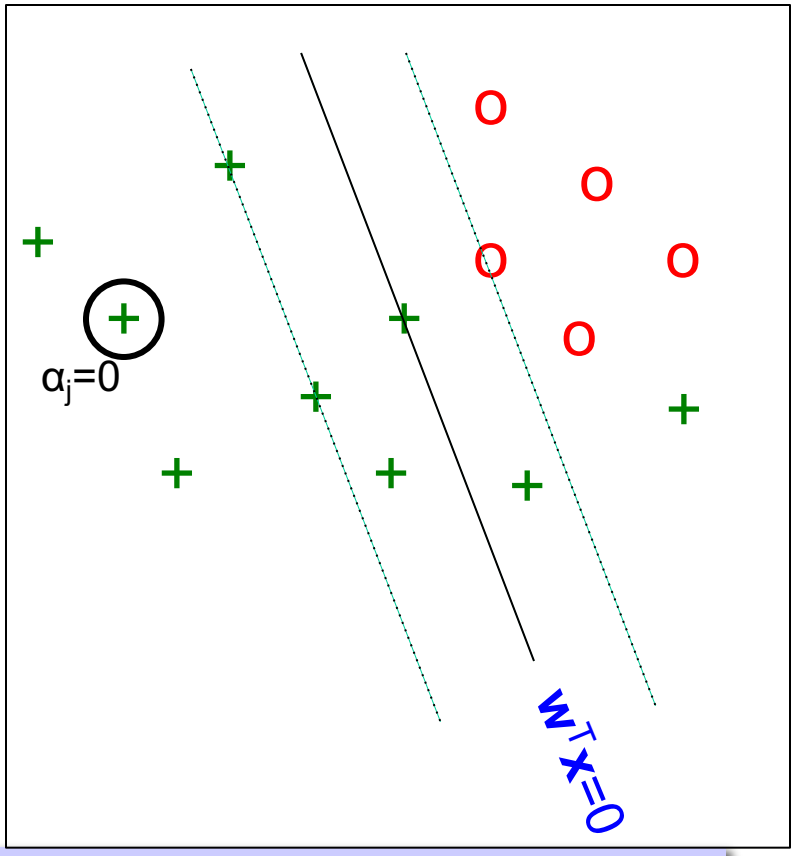
- Soft-SVM has one more constraint $0 \leq \alpha_i \leq C$ (vs $0 \leq \alpha_i$ in Hard SVM).
- When $C \Rightarrow \infty$, then Soft-SVM=>Hard-SVM.



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.

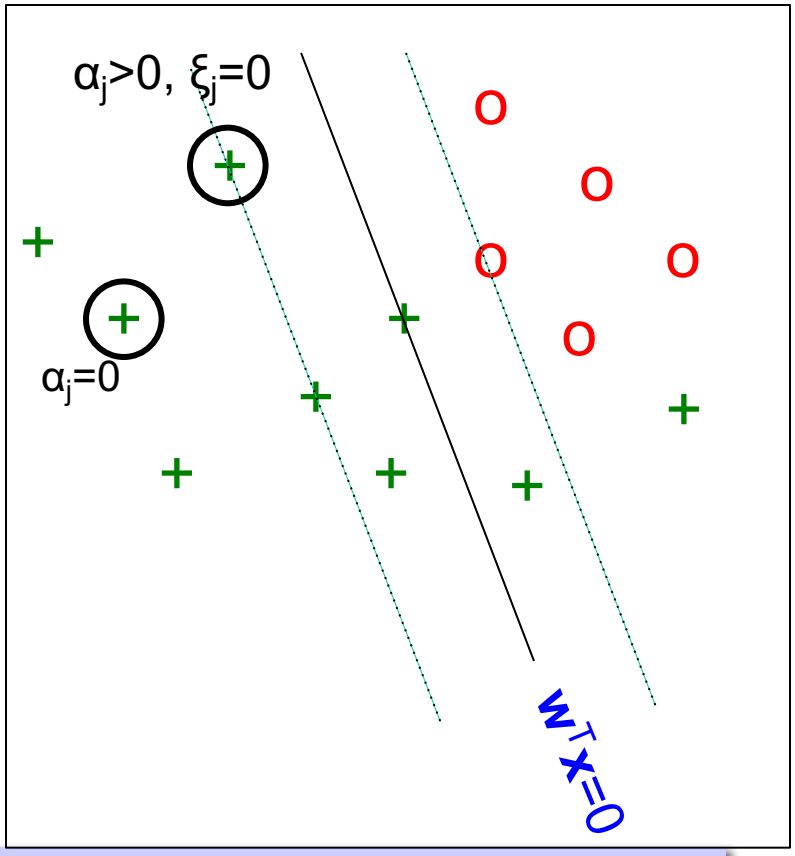
- Points away from margin have $\alpha_i = 0$, $\xi_i = 0$.
- The following have $\alpha_i > 0$
  - Points on the margin, $\xi_i = 0$.
  - Points within the margin, $0 < \xi_i < 1$
  - Points on the decision line, $\xi_i = 1$.
  - Misclassified points, $\xi_i > 1$.



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.

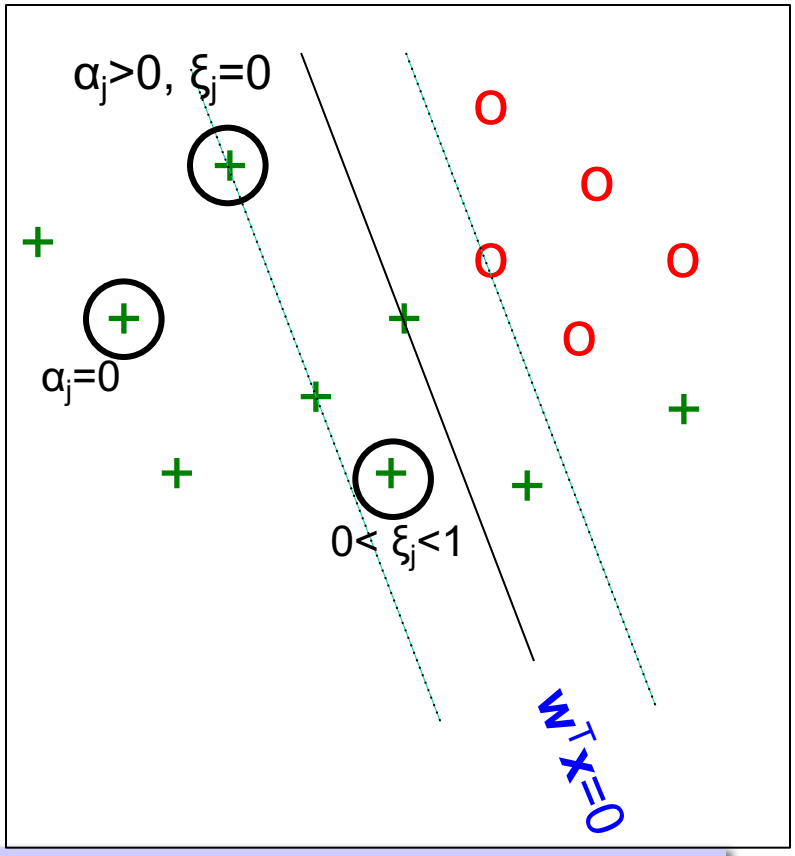
- Points away from margin have $\alpha_i = 0$, $\xi_i = 0$.
- The following have $\alpha_i > 0$
  - Points on the margin, $\xi_i = 0$.
  - Points within the margin, $0 < \xi_i < 1$
  - Points on the decision line, $\xi_i = 1$.
  - Misclassified points, $\xi_i > 1$.



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.

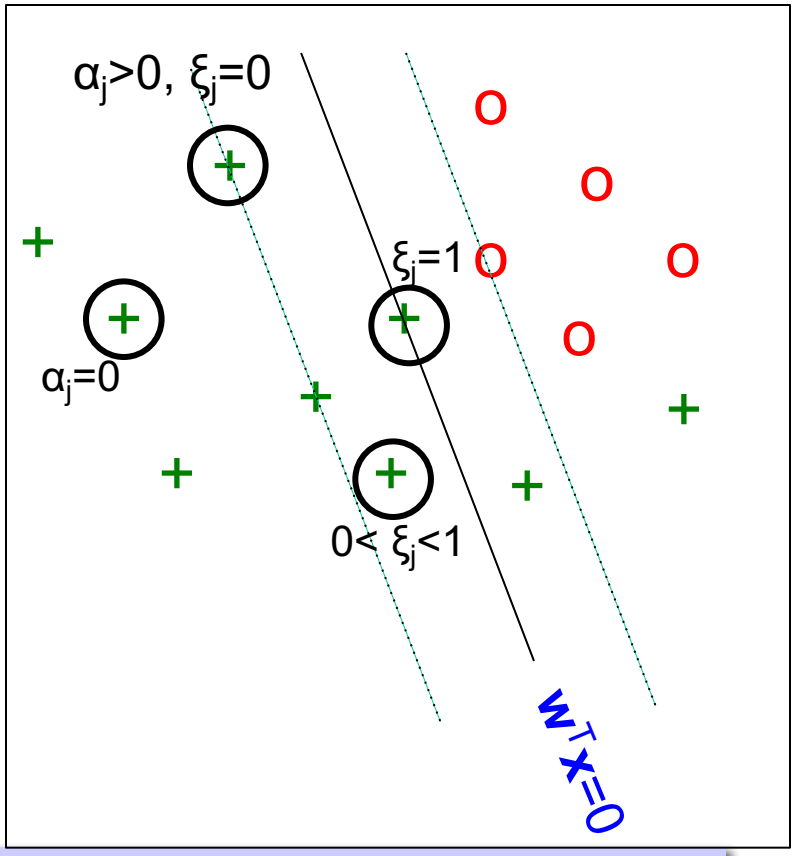
- Points away from margin have $\alpha_i = 0, \xi_i = 0$.
- The following have $\alpha_i > 0$
  - Points on the margin, $\xi_i = 0$.
  - Points within the margin, $0 < \xi_i < 1$
  - Points on the decision line, $\xi_i = 1$.
  - Misclassified points, $\xi_i > 1$.



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.

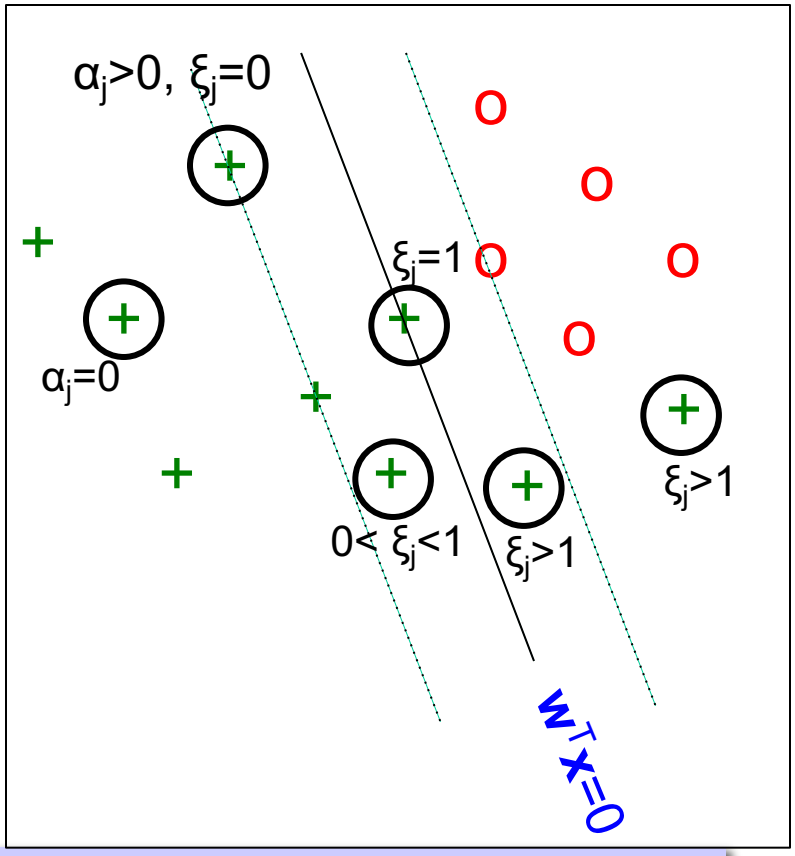
- Points away from margin have $\alpha_i = 0, \xi_i = 0$.
- The following have $\alpha_i > 0$
  - Points on the margin, $\xi_i = 0$.
  - Points within the margin, $0 < \xi_i < 1$
  - Points on the decision line, $\xi_i = 1$.
  - Misclassified points, $\xi_i > 1$.



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, $\xi_i > 1$.



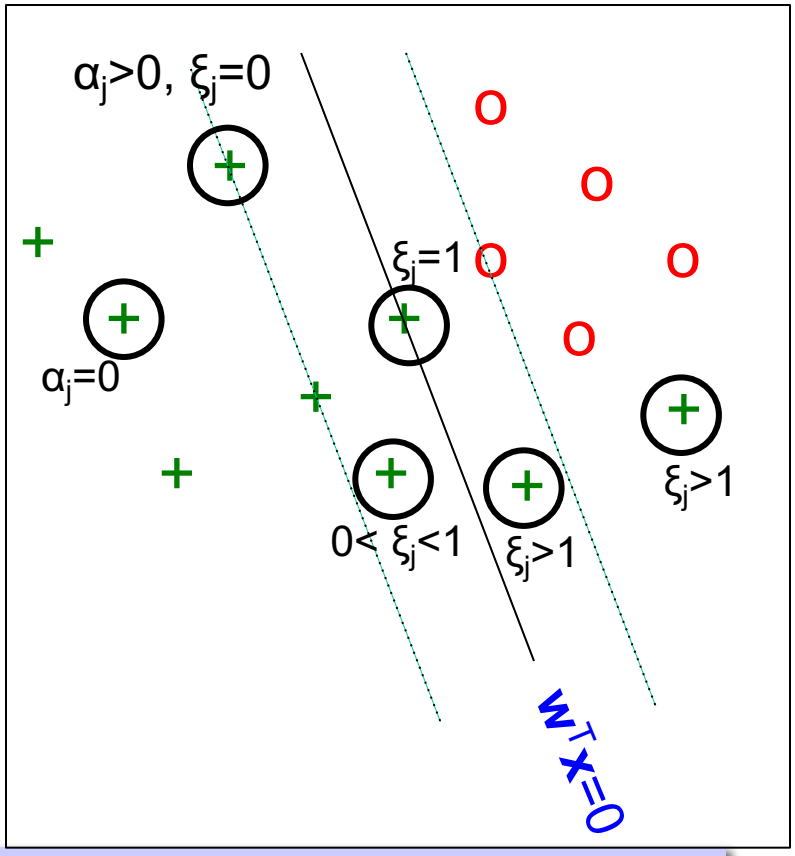
Soft SVM solution

- 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 $\alpha_i = 0, \xi_i = 0$.
- 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:

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

- Only need to store the support vectors (i.e. points with $\alpha_i > 0$) to predict.



Soft SVM, Bias & Variance

- $\hat{\mathbf{w}}_{soft} = argmin_{\mathbf{w},\xi}\ C \sum_{i:1:n} \xi_i + ½||\mathbf{w}||^2$
- $C \Rightarrow \infty$, 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 ϕ is called a feature mapping.

Margin optimization in feature space

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

  – Primal form:

$$\begin{aligned} &\textit{Min} && \tfrac{1}{2}\,||\boldsymbol{w}||^2 \\ &\text{w.r.t.} && \boldsymbol{w} \\ &\text{s.t.} && y_i \boldsymbol{w}^T \phi(\boldsymbol{x}_i) \geq 1 \end{aligned}$$

Margin optimization in feature space

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

  - Primal form:

    $$\begin{aligned} &\textit{Min} && \tfrac{1}{2}\,||\boldsymbol{w}||^2 \\ &\text{w.r.t.} && \boldsymbol{w} \\ &\text{s.t.} && y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i)+b) \geq 1 \end{aligned}$$

  - Dual form:

    $$\begin{aligned} &\textit{Max} && \textstyle\sum_i \alpha_i - \tfrac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j (\phi(\boldsymbol{x}_i)\cdot\phi(\boldsymbol{x})) \\ &\text{w.r.t.} && \alpha_i \\ &\text{s.t.} && \alpha_i \geq 0 \\ & && \textstyle\sum_i \alpha_i y_i = 0 \end{aligned}$$

Feature space solution

- The optimal weights, in the expanded 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:

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

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:

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

- 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 $K: R^m \times R^m \rightarrow R$, 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 $\phi(\mathbf{x}_1)\cdot\phi(\mathbf{x}_2) = |\phi(\mathbf{x}_1)|\cdot|\phi(\mathbf{x}_2)|\cdot\cos\angle(\phi(\mathbf{x}_1),\phi(\mathbf{x}_2))$, 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?

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

Example: Quadratic kernel

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

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

- 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?

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

- 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 m^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 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-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 just replace any inner product $\phi(\mathbf{x}_1)\cdot\phi(\mathbf{x}_2)$ with $K(\mathbf{x}_1, \mathbf{x}_2)$

- 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 α_i with the kernel trick

  (just replace $\phi(\mathbf{x}_1)\cdot\phi(\mathbf{x}_2)$ with $K(\mathbf{x}_1, \mathbf{x}_2)$)

$$\begin{array}{ll} \textit{Max} & \sum_{i=1:n} \alpha_i - \tfrac{1}{2} \sum_{i,j=1:n} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{w.r.t.} & \alpha_i \\ \text{s.t.} & \alpha_i \geq 0 \text{ and } \sum_{i:1..n} \alpha_i y_i = 0 \end{array}$$

- 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!

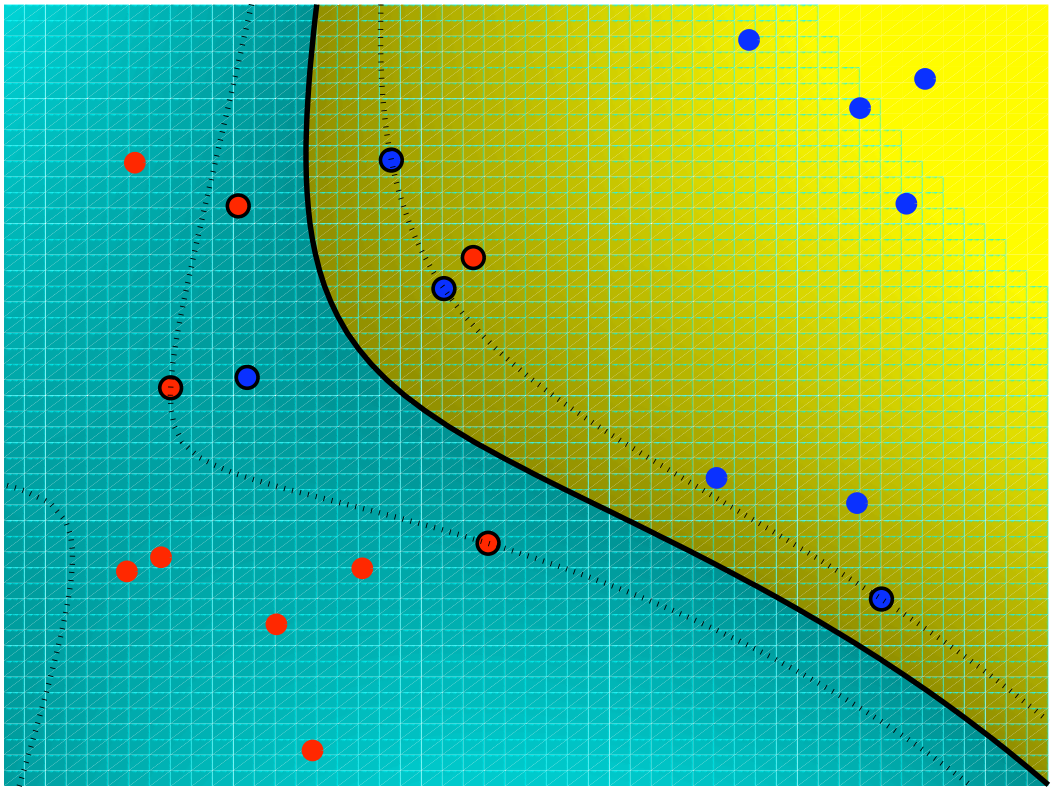
Some other kernel functions

- $K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}\cdot\mathbf{z})^d$ - 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=#original features)
  - It would be impossible to use this representation in a 'primal' formulation!

- 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

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.

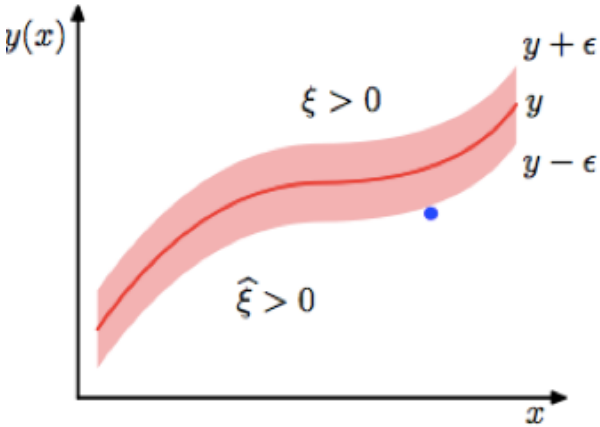
- Multi-class SVM: Define the margin to be the gap between the correct class and the nearest other class.

SVMs for regression

- Minimize a regularized error function:

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

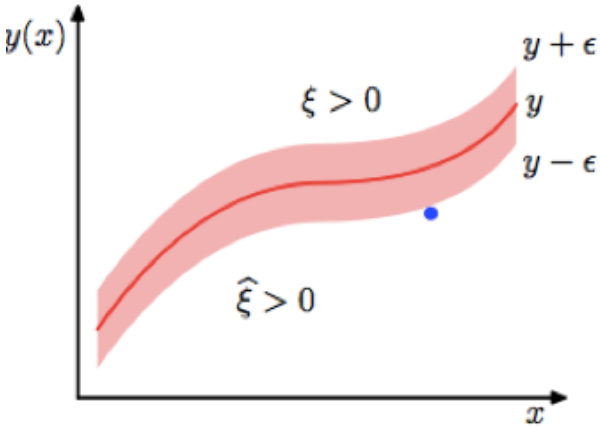
- Introduce slack variables to optimize "tube" around the regression function.



SVMs for regression

- Minimize a regularized error function:

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



- 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):

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

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

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.