CSCE 633: Machine Learning

Lecture 11: Support Vector Machines

Texas A&M University

9-18-19

Last Time

- Maximal Marginal Classifier
- Support Vector Classifier

Goals of this lecture

- Support Vector Machines - an overview

Maximal Marginal Hyperplane

- The maximal margin hyperplane depends directly on the points that lie on the margin
- These are called the support vectors
- So how do we build it?

$$x_1, \cdots, x_n \in \mathbb{R}^p$$
$$y_1, \cdots, y_n \in \{-1, +1\}$$

Then we want to:

$$maximize_{\beta_0, \beta_1, \cdots, \beta_p, M} M$$

Subject to constraints:

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

and

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M \forall i = 1, \cdots, n$$

Support Vector Classifier

$$x_1, \cdots, x_n \in \mathbb{R}^p$$

$$y_1, \cdots, y_n \in \{-1, +1\}$$

Then we want to:

$$maximize_{\beta_0, \beta_1, \cdots, \beta_p, M} M$$

Subject to constraints:

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

and

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

$$\forall i = 1, \cdots, n$$

Support Vector Classifier: Slack Variables

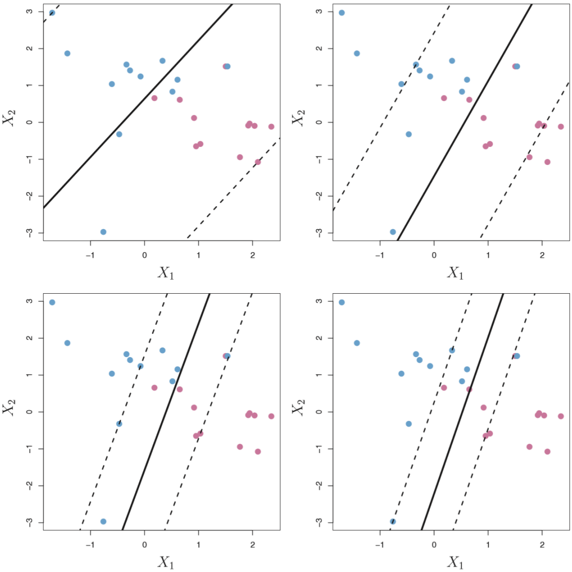
$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$
$$\forall i = 1, \cdots, n$$

where

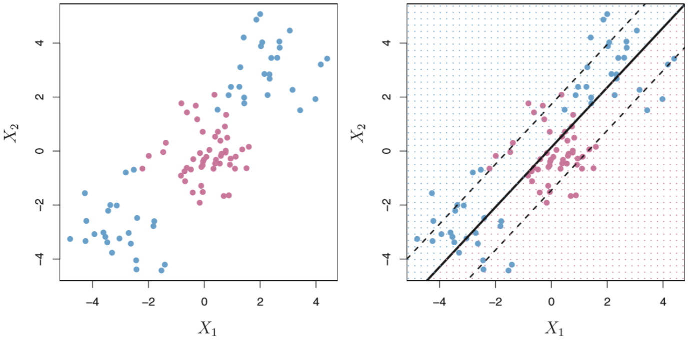
$$\epsilon_i \geq 0$$
$$\sum_{i=1}^{n} \epsilon_i \leq C$$

- C is a non-negative tuning parameter
- M is the width of the margin
- ϵ_i are the slack variables. When $\epsilon_i > 1$ the object is on the wrong side of the hyperplane, when $\epsilon_i > 0$ the object violates the margin
- Therefore, C determines the number and severity of margin violations

SVC



Non-separable



- SVC is natural for 2 class decision
- Remember back to Logistic Regression with interaction terms
- $x_1, x_2, \cdots, x_p, x_1^2, x_2^2, \cdots, x_p^2$ now we have p terms
- We can re-write SVC to maximize M subject to

$$y_i(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \geq M(1 - \epsilon_i)$$

and

$$\sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1$$

Can we enlarge the feature space even more? Would this give us non-linear decision boundaries?

Support Vector Machines: Enlarge through Kernel Space

- SVC is solved through the kernel space, with an inner product

Support Vector Machines: Enlarge through Kernel Space

- SVC is solved through the kernel space, with an inner product
- $< x_i, x_{i'} >= \sum_{j=1}^{p} x_{ij} x_{i'j}$
- Linear SVC finds $f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i >$

Support Vector Machines: Enlarge through Kernel Space

- SVC is solved through the kernel space, with an inner product
- $< x_i, x_{i'} >= \sum_{j=1}^{p} x_{ij} x_{i'j}$
- Linear SVC finds $f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i >$
- where n α_i exist per training example to estimate $\alpha_1, \cdots, \alpha_n$ and β

Support Vector Machines: Enlarge through Kernel Space

- SVC is solved through the kernel space, with an inner product
- $< x_i, x_{i'} >= \sum_{j=1}^{p} x_{ij} x_{i'j}$
- Linear SVC finds $f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i >$
- where n α_i exist per training example to estimate $\alpha_1, \cdots, \alpha_n$ and β
- For this we need $\binom{n}{2}$ inner products

- SVC is solved through the kernel space, with an inner product
- $< x_i, x_{i'} >= \sum_{j=1}^{p} x_{ij} x_{i'j}$
- Linear SVC finds $f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i >$
- where n α_i exist per training example to estimate $\alpha_1, \cdots, \alpha_n$ and β
- For this we need $\binom{n}{2}$ inner products
- $\alpha_i \neq 0$ only for the support vectors, so we can re-write as

Support Vector Machines: Enlarge through Kernel Space

- SVC is solved through the kernel space, with an inner product
- $< x_i, x_{i'} >= \sum_{j=1}^{p} x_{ij} x_{i'j}$
- Linear SVC finds $f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i >$
- where n α_i exist per training example to estimate $\alpha_1, \cdots, \alpha_n$ and β
- For this we need $\binom{n}{2}$ inner products
- $\alpha_i \neq 0$ only for the support vectors, so we can re-write as
- $f(x) = \beta_0 + \sum_{i \in S} \alpha_i < x, x_i >$

Support Vector Machines: Higher Dimensions

- In higher dimensions, we don't need to know the full feature space, just how to calculate the inner product $K(x_i, x_{i'})$
- K is a kernel - it quantifies similarity of two observations
- when $K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$ we get linear SVC. This is called the linear kernel.
- Similarly we can define $K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$ as the polynomial kernel of degree d, which is much more flexible with the decision boundary but needs more data to train.

SVM, then, defines:

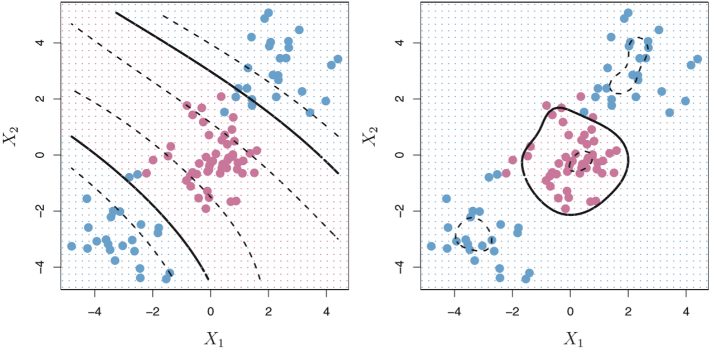
$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i)$$

Support Vector Machines: Radial Kernel

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

- This is a popular kernel, requires even more data to train
- It has very local behavior, because of the sum of squared difference
- Rather than simply adding additional features, using these kernels is better computationally
- Feature space is implicit and infinite-dimensional so we could never compute the full model in those spaces

RBF Decision Boundary



Understanding the Loss Function

- Consider the margin boundary $y_i \cdot f(x) = y_i(\beta_0 + \sum_{j=1}^{p} \beta_j \phi(x_{ij}))$

Understanding the Loss Function

- Consider the margin boundary $y_i \cdot f(x) = y_i(\beta_0 + \sum_{j=1}^{p} \beta_j \phi(x_{ij}))$
- Why do we multiply y by $f(x)$?

Understanding the Loss Function

- Consider the margin boundary $y_i \cdot f(x) = y_i(\beta_0 + \sum_{j=1}^{p} \beta_j \phi(x_{ij}))$
- Why do we multiply y by $f(x)$?
- Can we somehow relate SVM's margin boundary to Regression's Loss Functions?

Understanding the Loss Function

- Consider the margin boundary $y_i \cdot f(x) = y_i(\beta_0 + \sum_{j=1}^{p} \beta_j \phi(x_{ij}))$
- Why do we multiply y by $f(x)$?
- Can we somehow relate SVM's margin boundary to Regression's Loss Functions?
- Recall $y - \hat{y} = y - f(x)$ is our residual (error)

Classification Rule

- The classification rule for SVM is $G(x) = sgn(f(x))$

Classification Rule

- The classification rule for SVM is $G(x) = sgn(f(x))$
- Now, how do we classify error?

Classification Rule

- The classification rule for SVM is $G(x) = sgn(f(x))$
- Now, how do we classify error?
- Recall $y_i \in \{-1, +1\}$

Classification Rule

- The classification rule for SVM is $G(x) = sgn(f(x))$
- Now, how do we classify error?
- Recall $y_i \in \{-1, +1\}$
- So, $y_i \cdot G(x_i) > 0$ if samples are classified correctly

Classification Rule

- The classification rule for SVM is $G(x) = sgn(f(x))$
- Now, how do we classify error?
- Recall $y_i \in \{-1, +1\}$
- So, $y_i \cdot G(x_i) > 0$ if samples are classified correctly
- Why? two cases: $y_i = -1$ and $G(x_i) = -1$ or $y_i = +1$ and $G(x_i) = +1$

0-1 Loss

- If we define a decision boundary as $f(x) = 0$, then more generally

0-1 Loss

- If we define a decision boundary as $f(x) = 0$, then more generally
- $L(y, f(x))$ is called the 0-1 loss in this case

0-1 Loss

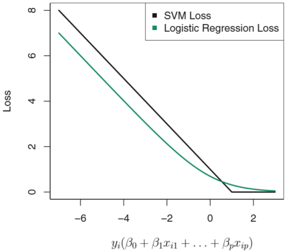
- If we define a decision boundary as $f(x) = 0$, then more generally
- $L(y, f(x))$ is called the 0-1 loss in this case
- $L(y, f(x)) = \mathbb{I}(y \cdot f(x) < 0)$

Support Vector Machines: Hinge Loss

$$L(X, y, \beta) = \sum_{i=1}^{n} \max[0, 1 - y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})]$$

instead of the common loss for logistic regression

$$L(X, y, \beta) = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$

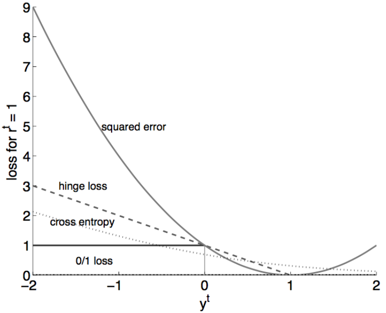


Support Vector Machines: Hinge Loss

$$L(X, y, \beta) = \sum_{i=1}^{n} \max[0, 1 - y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})]$$

instead of the common loss for logistic regression

$$L(X, y, \beta) = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$



Optimal Hyperplane and Support Vectors

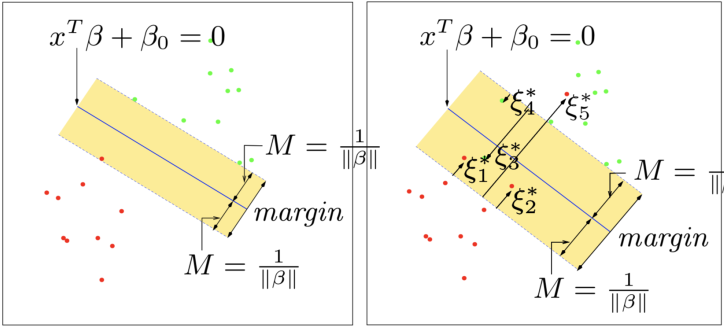


FIGURE 12.1. *Support vector classifiers. The left*

- **Margin of separation** M: distance between the separating hyperplane and the closest input point.
- **Support vectors**: input points closest to the separating hyperplane.

Turn a constrained optimization problem into an unconstrained optimization problem by absorbing the constraints into the cost function, weighted by the *Lagrange multipliers*

Example: Find point on the circle $x^2 + y^2 = 1$ closest to the point $(2, 3)$

- Minimize $F(x, y) = (x-2)^2 + (y-3)^2$ subject to the constraint $x^2 + y^2 - 1 = 0$.
- Absorb the constraint into the cost function, after multiplying the *Lagrange multiplier* $\alpha > 0$:

$$F(x, y, \alpha) = (x-2)^2 + (y-3)^2 + \alpha(x^2 + y^2 - 1).$$

Formulate Lagrangian (primal problem):
$F(x, y, \alpha) = (x-2)^2 + (y-2)^2 + \alpha(x^2 + y^2 - 1)$, with $\alpha > 0$
The optimization problem becomes $\max_\alpha \min_{x,y} F(x, y, \alpha)$
Minimize $\min_{x,y} F(x, y, \alpha)$

$$\frac{\partial F}{\partial x} = 2(x-2) + 2\alpha x = 0 \Rightarrow x = \frac{2}{1+\alpha}$$

$$\frac{\partial F}{\partial y} = 2(y-2) + 2\alpha y = 0 \Rightarrow y = \frac{2}{1+\alpha}$$

We substitute x, y in the Lagrangian and express it in terms of its dual from wrt α and maximize it

$$\frac{\partial F}{\partial \alpha} = x^2 + y^2 - 1 = 0 \Rightarrow \left(\frac{2}{1+\alpha}\right)^2 + \left(\frac{2}{1+\alpha}\right)^2 = 1 \Rightarrow \alpha = 2\sqrt{2} - 1$$

Recover the solution for the x, y $(x, y) = (1/\sqrt{2}, 1/\sqrt{2})$

Primal Problem: Constrained Optimization

Linearly separable case

For the training set $\mathcal{D}^{train} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ find β and β_0 such that

- they minimize the *inverse* separation margin ($\frac{1}{M} = \frac{\|\beta\|}{2}$) while satisfying a constraint (all examples are correctly classified):
  - Cost function: $\Phi(\beta) = \frac{1}{2}\beta^T\beta$
  - Constraint: $y_n(\beta^T\mathbf{x}_i + b) \geq 1$ for $i = 1, 2, ..., N$

$$\boxed{\min \frac{1}{2}\|\beta\|_2^2, \text{ such that (s.t.) } y_n(\beta^T\mathbf{x} + \beta_0) \geq 1 \,, \quad n = 1, \ldots, N}$$

This problem can be solved using the *method of Lagrange multipliers* (see next two slides)

Support Vector Machines: Linearly separable case

$$\boxed{\min \tfrac{1}{2}\|\beta\|_2^2, \ \text{such that (s.t.)} \ y_n(\beta^T \mathbf{x} + \beta_0) \geq 1 \ , \ \ n = 1, \dots, N}$$

1) Formulate Lagrangian function (primal problem)

$L_p = \frac{1}{2}\|\beta\|_2^2 - \sum_{n=1}^N \alpha_n \left[y_n(\beta^T \mathbf{x}_n + \beta_0) - 1\right]$

2) Minimize Lagrangian to solve for primal variables β, β_0

$\frac{\partial L_p}{\partial \beta} = 0 \ \ \Rightarrow \ \ \beta = \sum_{n=1}^N \alpha_n y_n \mathbf{x_n}$

$\frac{\partial L_p}{\partial \beta_0} = 0 \ \ \Rightarrow \ \ \sum_{n=1}^N \alpha_n y_n = 0$

3) Substitute the primal variables β, β_0 into the Lagrangian and express in terms of dual variables α_n

$$L_d = \frac{1}{2}\|\beta\|_2^2 - \beta^T \sum_{n=1}^N \alpha_n y_n \mathbf{x_n} - \beta_0 \sum_{n=1}^N \alpha_n y_n + \sum_{n=1}^N \alpha_n$$

$$= -\frac{1}{2}\beta^T \beta + \sum_{n=1}^N \alpha_n = -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^M \alpha_n \alpha_m y_n y_m \mathbf{x_n}^T \mathbf{x_m} + \sum_{n=1}^N \alpha_n$$

$$\min \tfrac{1}{2}\|\beta\|_2^2, \;\text{ such that (s.t.) }\; y_n(\beta^T \mathbf{x} + \beta_0) \geq 1 \;, \quad n = 1, \dots, N$$

4) Maximize the Lagrangian with respect to dual variables (dual problem)

$$\max_{\alpha_n} L_d = \max_{\alpha_n} \left\{ -\frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} \alpha_n \alpha_m y_n y_m \mathbf{x_n}^T \mathbf{x_m} + \sum_{n=1}^{N} \alpha_n \right\}$$

s.t. $\sum_{n=1}^{N} \alpha_n y_n = 0$ and $\alpha_n \geq 0$, for $n = 1, \dots, N$

- Solved numerically using quadratic optimization methods
- The dual depends on data size N and not on data dimensionality D
- Most of the α_n will vanish with $\alpha_n = 0$, only a small percentage will have $\alpha_n > 0$
- The set of $\mathbf{x_n}$ whose $\alpha_n > 0$ are the support vectors

$$\boxed{\min \tfrac{1}{2}\|\beta\|_2^2, \text{ such that (s.t.) } y_n(\beta^T \mathbf{x} + \beta_0) \geq 1\ , \quad n = 1, \ldots, N}$$

5) Recover the solution (for the primal variables) from the dual variables

- Find β: Substitute α_n from (4) to $\beta = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x_n}$
- Find β_0:
  - From $\beta^T \mathbf{x_n} + \beta_0 = y_n$, where $\mathbf{x_n}$ is a support vector, calculate $\beta_0 = y_n - \beta^T \mathbf{x_n}$.
  - For numerical stability average β_0 values estimated from all support vectors.

Testing

- Testing doesn't enforce a margin, i.e. $g(\mathbf{x}) = \beta^T \mathbf{x} + \beta_0$
- Choose C_1 if $g(\mathbf{x}) > 0$, C_2 otherwise
- Function $g(\mathbf{x_n}) = \beta^T \mathbf{x} + \beta_0 = \sum_{m=1}^{N} \alpha_m y_m \mathbf{x_m}^T \mathbf{x_n} + \beta_0$ does not need explicit calculation of β
  - $g(\mathbf{x_n})$ be calculated from the dot product between the input vectors $\mathbf{x_m}^T \mathbf{x_n}$ (*keep this in mind for later*)

Support Vector Machines: Linearly separable case

- Samples $\mathbf{x_n}$ for which $\alpha_n = 0$
  - majority of samples
  - lie away from the hyperplane: $y_n(\beta^T \mathbf{x_n} + \beta_0) > 1$
  - have no effect on the hyperplane
- Samples $\mathbf{x_n}$ for which $\alpha_n = 0$
  - support vectors
  - lie close to the hyperplane: $y_n(\beta^T \mathbf{x_n} + \beta_0) = 1$
  - determine the hyperplane

- If two classes are not linearly separable, we look for the hyperplane that yields the least error
- We define slack variables $\xi_n \geq 0$ which represent the deviation from the margin

$$y_n(\beta^T \mathbf{x_n} + \beta_0) \geq 1 - \xi_n$$

- *Case (a):* Far away from the margin, $\xi_n = 0$
- *Case (b):* On the right side and far from margin, $\xi_n = 0$
- *Case (c):* On the right side, but in the margin, $0 \leq \xi_n \leq 1$
- *Case (d):* On the wrong side, $\xi_n \geq 1$

We incorporate the number of misclassifications $\#\{\xi_n > 1\}$ and the number of non-separable points $\#\{\xi_n > 0\}$ as a soft error $\sum_n \xi_n$.

Support Vector Machines: Non-separable case

$$\boxed{\min \tfrac{1}{2}\|\beta\|_2^2 + C\sum_{n=1}^{N}\xi_n, \;\; \text{s.t. } \; y_n(\beta^T\mathbf{x}+\beta_0) \geq 1-\xi_n \;\text{ and }\; \xi_n \geq 0\,, \;\; \forall n}$$

1) Formulate Lagrangian function (primal problem)

$$L_p = \tfrac{1}{2}\|\beta\|_2^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n\left[y_n(\beta^T\mathbf{x}_n+\beta_0) - 1 + \xi_n\right] - \sum_{n=1}^{N}\mu_n\xi_n$$

2) Minimize Lagrangian to solve for primal variables β, β_0

$$\frac{\partial L_p}{\partial \beta} = 0 \;\;\Rightarrow\;\; \beta = \sum_{n=1}^{N}\alpha_n y_n \mathbf{x_n}$$

$$\frac{\partial L_p}{\partial \beta_0} = 0 \;\;\Rightarrow\;\; \sum_{n=1}^{N}\alpha_n y_n = 0$$

$$\frac{\partial L_p}{\partial \xi_n} = 0 \;\;\Rightarrow\;\; C - \alpha_n - \mu_n = 0$$

Support Vector Machines: Non-separable case

3) Substitute the primal variables β, β_0 into the Lagrangian and express in terms of dual variables α_n

$$
\begin{aligned}
L_d &= \frac{1}{2}\|\beta\|_2^2 + C\sum_{n=1}^{N}\xi_n - \beta^T\sum_{n=1}^{N}\alpha_n y_n \mathbf{x_n} \\
&- \beta_0\sum_{n=1}^{N}\alpha_n y_n + \sum_{n=1}^{N}\alpha_n - \sum_{n=1}^{N}\alpha_n\xi_n - \sum_{n=1}^{N}\mu_n\xi_n \\
&= \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M}\alpha_n\alpha_m y_n y_m \mathbf{x_n}^T\mathbf{x_m} + \sum_{n=1}^{N}\xi_n\left(C - \alpha_n - \mu_n\right) - \\
&- \sum_{m=1}^{N}\alpha_m y_m \mathbf{x_m}^T \cdot \sum_{n=1}^{N}\alpha_n y_n \mathbf{x_n} - \beta_0\sum_{n=1}^{N}\alpha_n y_n + \sum_{n=1}^{N}\alpha_n \\
&= \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M}\alpha_n\alpha_m y_n y_m \mathbf{x_n}^T\mathbf{x_m}
\end{aligned}
$$

Support Vector Machines: Non-separable case

4) Maximize the Lagrangian with respect to dual variables (dual problem)

$$\max_{\alpha_n} \left\{ \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} \alpha_n \alpha_m y_n y_m \mathbf{x_n}^T \mathbf{x_m} \right\}$$

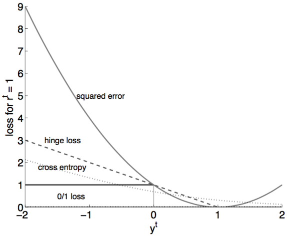
s.t. $\sum_{n=1}^{N} \alpha_n y_n = 0$ and $0 \leq \alpha_n \leq C$, for $n = 1, \ldots, N$

- Solved numerically using quadratic optimization methods
- $\alpha_n = 0$: instances at the correct side of the hyperplane with sufficient margin
- $\alpha_n > 0$: support vectors
    - $0 < \alpha_n < C$: instances lying on the margin
    - $\alpha_n = C$: instances in the margin or misclassified

Support Vector Machines: Hinge Loss

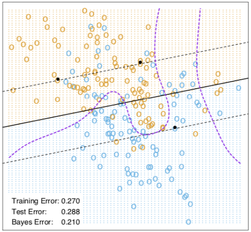
- Decision rule: $f(\mathbf{x}) = \text{sign}(\beta^T\mathbf{x} + \beta_0)$
    - $f(\mathbf{x}) = 1$, if $\beta^T\mathbf{x} + \beta_0 > 0$
    - $f(\mathbf{x}) = -1$, if $\beta^T\mathbf{x} + \beta_0 < 0$
- If $f(\mathbf{x})$ is the output and y_n the actual label

$$l^{\text{hinge}}(f(\mathbf{x}), y) = \begin{cases} 0 & \text{if } y(\beta^T\mathbf{x} + \beta_0) \geq 1 \\ 1 - y(\beta^T\mathbf{x} + \beta_0) & \text{otherwise} \end{cases}$$

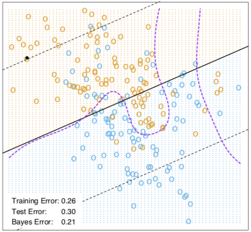


Intuition: penalize more if incorrectly classified

Support Vector Machines: Tuning C



$C = 10000$



$C = 0.01$

So far

- SVM aims at finding the hyperplane from which instances have a margin of distance
- Prime and dual problem formulation (Lagrange multiplies)
- Support vectors: instances closest to separating hyperplane
- Linearly separable case: maximize margin of separation between two classes
- Non-separable case: look for the hyperplane that yields the least error (soft error)
    - Prime: minimizes Lagrangian wrt the primal variables of the problem
    - Dual: maximizes Lagrangian wrt multipliers

Takeaways and Next Time

- Support Vector Machines
- Next Time: Support Vector Machines - Kernels