Computational Data Analysis

The Support Vector Machine and Convex Optimization

Lars Arvastson and Line Clemmensen

February 28, 2018

Todays Lecture

- Recap
- Convex optimization using Lagrange multipliers
- Optimal Separating Hyperplanes
- Support Vector Machines
- The kernel trick

Last Week

- Linear discriminant analysis and Logistic regression
  - What for?
  - How do they compare?

- Basis expansion
  - What is it?
  - How did we use it?

Crash course in constrained optimization

We learn to solve $\begin{cases} \max_x f(x) \\ g(x) = 0 \\ h(x) \geq 0 \end{cases}$

using Lagrange multipliers

Why?
Because we will use it to build the Support Vector Machine!

Unconstrained optimization

Solve

$$\max_x f(x)$$

Assume that f is nice, i.e. continuously differentiable.

Then a local maxima, x^* fulfills

1. Gradient is zero, $\nabla_x f(x^*) = 0$
2. Hessian is negative definite, $v^T \nabla^2_{xx} f(x^*) v < 0, \forall v \in \mathrm{R}^n$

where

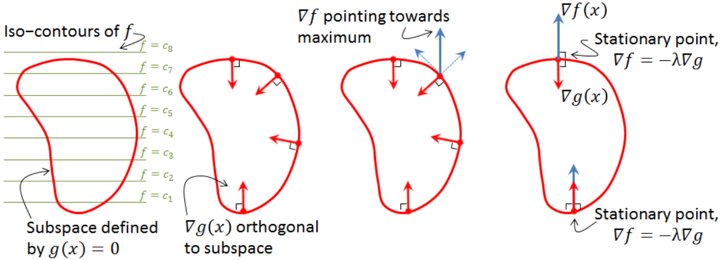
$$\nabla_x f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix} \qquad \nabla^2_{xx} f(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}$$

A negative second derivative guarantees a **local maximum** (otherwise saddle point or local minimum).

Constrained optimization

Now, assume that any x is not good enough. Introduce a constraint that x must fullfill,

$$\begin{cases} \max_x f(x) \\ g(x) = 0 \end{cases}$$



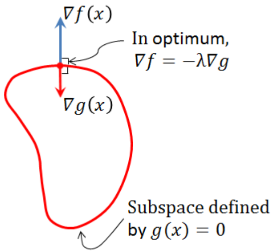
- The stationary points are defined by $\nabla f = -\lambda \nabla g$ for some constant λ

Lagrange multipliers

Define the **Lagrange primal function**

$$L_p(x, \lambda) = f(x) + \lambda g(x)$$

and the **Lagrange multiplier** λ.



Find solution (x^*, λ^*) to

$$\max_x \min_\lambda L_P(x, \lambda)$$

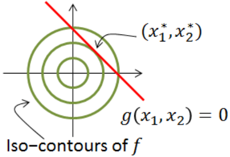
by solving

$$\begin{cases} \frac{\partial L_p}{\partial x} & = 0 \\ \frac{\partial L_p}{\partial \lambda} & = 0 \end{cases} \quad \text{i.e. } \nabla L_p = 0.$$

The stationary points, x^*, might be local maxima, local minima or saddle points. Verify that the Hessian is negative semi-definite.

Example

$$\begin{cases} \max_x f(x_1, x_2) = 1 - x_1^2 - x_2^2 \\ g(x_1, x_2) = x_1 + x_2 - 1 = 0 \end{cases}$$



$$\begin{aligned} L_P(\mathbf{x}, \lambda) &= f(x) + \lambda g(x) \\ &= 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1) \end{aligned}$$

$$\begin{cases} \frac{\partial L_p}{\partial x_1} & = -2x_1 + \lambda = 0 \\ \frac{\partial L_p}{\partial x_2} & = -2x_2 + \lambda = 0 \\ \frac{\partial L_p}{\partial \lambda} & = x_1 + x_2 - 1 = 0 \end{cases}$$

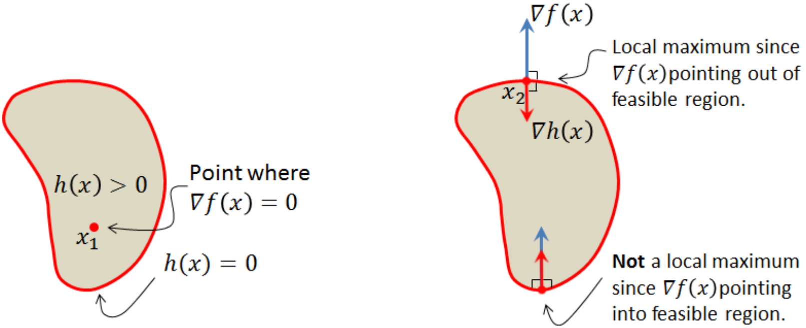
Solution/optimum is at
$(x_1^*, x_2^*) = (1/2, 1/2)$ with $\lambda = 1$.

Inequalities in constraints, $h(x) \geq 0$

Constrained optimization with inequality constraints

$$\begin{cases} \max_x f(x) \\ h(x) \geq 0 \end{cases}$$

The optimum is either within the feasible region $h(x) \geq 0$ or along the edge $h(x) = 0$.



Notice that ∇h is always pointing inwards since $h > 0$ in the feasible region and $h = 0$ along the edge.

Inequalities in constraints, $h(x) \geq 0$

1. Constraint is inactive, ie $h(x_1) > 0$.
   - Solution by $\nabla f(x_1) = 0$, ie Lagrange function,

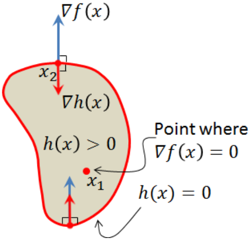
$$\begin{cases} \nabla f(x_1) = -\mu \nabla h(x_1) \\ \mu = 0 \end{cases}$$

   - Notice that $\mu h(x_1) = 0$, since $\mu = 0$
   - Maximum if negative definite Hessian

2. Constraint is active, ie $h(x_2) = 0$.
   - As before with $\mu \neq 0$. Important with the sign of μ. In maximum $\nabla f(x_2)$ is pointing out of the region $h(x_2) > 0$, ie

$$\begin{cases} \nabla f(x_2) = -\mu \nabla h(x_2) \\ \mu > 0 \end{cases}$$

   - Notice that $\mu h(x_1) = 0$, since $h(x_2) = 0$
   - Maximum if negative semidefinite Hessian



Lagranges problem for inequality constraints

A local maximum to the constrained optimization problem

$$\begin{cases} \max_x f(x) \\ h(x) \geq 0 \end{cases}$$

with Lagrange function

$$L_p(x, \mu) = f(x) + \mu h(x)$$

is given by (x^*, μ^*) when (Karush-Kuhn-Tucker conditions)

1. $\nabla_x L_P(x^*, \mu^*) = 0$
2. $\mu^* \geq 0$
3. $\mu^* h(x^*) = 0$
4. $h(x^*) \geq 0$
5. Negative definite constraints on Hessian

For a minimization problem we change sign, $L_p(x, \mu) = f(x) - \mu h(x)$.

Multiple constraints

Multiple constraints,

$$\begin{cases} \max_x f(x) \\ g_j(x) = 0 \quad \forall j \\ h_k(x) \geq 0 \quad \forall k \end{cases}$$

are handle with more Lagrange multipliers,

$$L_p(x, \lambda, \mu) = f(x) + \sum_j \lambda_j g_j(x) + \sum_k \mu_k h_k(x).$$

Lagrange dual problem

The **Lagrange primal** problem is

$$\max_{x} \min_{\substack{\lambda \\ \mu \geq 0}} L_P(x, \lambda, \mu)$$

If we swap the order of min and max we get the **Lagrange dual** problem,

$$\min_{\substack{\lambda \\ \mu \geq 0}} \max_{x} L_P(x, \lambda, \mu)$$

Often these two problems have the same solution.

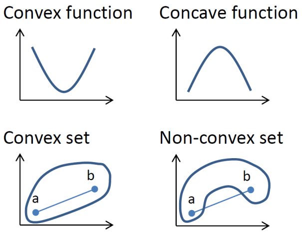
Define **Lagrange dual function**

$$L_D(\lambda, \mu) = \max_{x} L_P(x, \lambda, \mu)$$

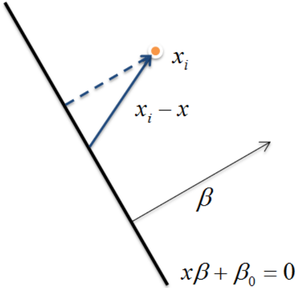
Slater's condition

> **Slater's condition**
> The primal and dual optimization problems are equivalent when f is concave and constraints are convex.

- There must be some x fulfilling all constraints
- Linear constraints are OK
- A local optimum will also be the global optimum.
- Not necessary to check conditions on the Hessian.



Example -
Shortest distance from point to line



$$\begin{cases} \arg\min_x \frac{1}{2}(x_i - x)(x_i - x)^T \\ \text{such that} \\ \quad x\beta + \beta_0 = 0 \end{cases}$$

Solve using Lagrange primal function

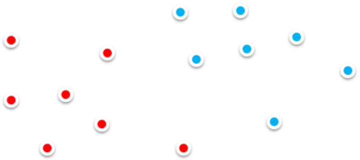
Optimal Separating Hyperplane

- Binary classification
- Sometimes data are perfectly separated by a straight line
- No overlap, one class on one side and the other class on the other side
- Not very useful in practice but it can be modified into the powerful Support Vector Machine

The decision function

Linear decision functions

$$y_{\text{new}} = \text{sign}(x_{\text{new}}\beta + \beta_0)$$

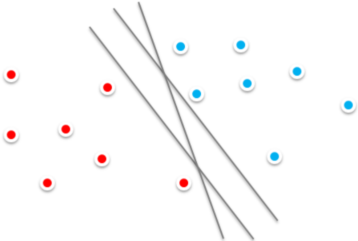


- For practical reason we label the two classes 1 and -1.
- Fitting the model involves choosing values for β and β_0
- Binary classification, extensions can be made
  - One vs. the rest
  - One vs. one

  both approaches uses several models.

The decision function

- Many hyperplanes can separate the two classes
- What would be optimal?



Linear Discriminant Analysis,
used all data to define Σ and μ from which the decision line was derived.

Logistic regression,
defined decision line emphasizing data close to line.

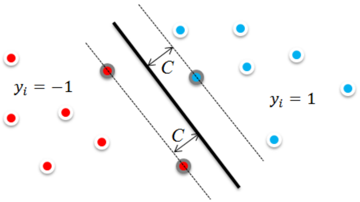
Optimal Separating Hyperplanes,
goes to the extreme and defines decision line based on closest observations only.

Introduce the margin

Maximize the distance *C* from the decision line to the nearest points in each class.

There is **no probabilistic model** here as we have for linear discriminant analysis and logistic regression

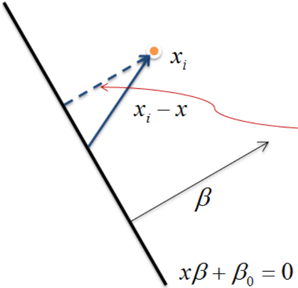
Hence, no probability for class belonging and no ML-estimation



Distance from point to plane

We wish to maximize the margin between classes.

We need an expression for point-to-plane distance



$$\frac{(x_i - x)\beta}{\|\beta\|} = \frac{x_i\beta + \beta_0}{\|\beta\|}$$

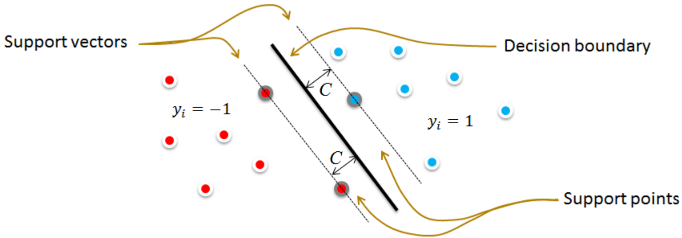
OSH as a maximization problem

We can now formulate a maximization problem

$$\arg\max_{\beta,\beta_0} C$$

such that

$$y_i \frac{x_i\beta + \beta_0}{||\beta||} \geq C \quad \forall i$$



The margin

Let x_+ be a support point in class 1 and x_- be a support point in class -1. Then

$$C = \frac{1}{2}\frac{\beta^T}{||\beta||}(x_+ - x_-)$$

Why?

Margin C is invariant to length of β. Choose length of β such that

$$\beta^T x_+ + \beta_0 = 1$$

$$\beta^T x_- + \beta_0 = -1$$

which gives

$$\beta^T(x_+ - x_-) = 2$$

The margin becomes

$$C = \frac{1}{||\beta||}$$

and the constraints simplifies into

$$y_i(x_i\beta - \beta_0) \geq 1 \quad \forall i$$

Solving the OSH problem

- Maximization problem can be turned into a minimization problem

$$\begin{cases} \quad \arg\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 \\ \text{such that} \\ \quad y_i(x_i\beta + \beta_0) \geq 1 \quad \forall i \end{cases}$$

- This is a nonlinear problem with **linear** constraints
  - You could use Matlabs `fmincon` function for constrained optimization of **any** nonlinear function
  - But this one is quadratic (convex) - does this simplify things?
- Quadratic programming
  - Very efficient solvers exists
  - Matlabs `quadprog`

Dual formulation of OSH problem

We have formulated the OSH problem such that we can use efficient standard numerical solvers. **We have**

- A model with one β coefficient for each dimension of x.
- One constraint for each observation x
- An optimal linear separation between classes

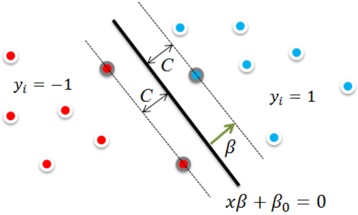
What else could we ask for? Well, it would be nice with

- One coefficient for each observation instead of each dimension.
  - Good idea for high-dimensional problems with few observations.
- A non-linear separation between classes.

We can achieve this if we use **Lagrange multipliers**

Solving the OSH problem

Use the Lagrange multipliers!



$$\begin{cases} \quad \arg\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 \\ \text{such that} \\ \quad y_i(x_i\beta + \beta_0) \geq 1 \quad \forall i \end{cases}$$

Step 1 Incorporate constraints using Lagrange multipliers, α_i,

$$\begin{cases} L(\beta, \beta_0, \alpha) = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{n} \alpha_i(y_i(x_\beta + \beta_0) - 1) \\ \alpha_i \geq 0 \quad \forall i \end{cases}$$

Solving the OSH problem, cont'd

Step 2 Differentiate and set to zero. This solves $\arg\min_{\beta,\beta_0} L_p$ (Lagrange dual),

$$\begin{cases} \frac{\partial L}{\partial \beta} = \beta - \sum_i \alpha_i y_i x_i^T = 0 \\ \frac{\partial L}{\partial \beta_0} = \sum \alpha_i y_i = 0 \end{cases}$$

and we have

$$\begin{cases} \beta = \sum_i \alpha_i y_i x_i^T \\ \sum \alpha_i y_i = 0 \end{cases}$$

OSH dual formulation

Step 3 Plug into original problem and simplify

$$
\begin{aligned}
L_D &= \frac{1}{2}|| \sum \alpha_i y_i x_i^T ||^2 - \sum (\alpha_i y_i (x_i \beta + \beta_0) - \alpha_i) \\
&= \dots \\
&= \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i x_j^T \\
&= \alpha \mathbf{1} - \frac{1}{2} \alpha^T Y X X^T Y \alpha \quad \text{where } Y = \mathrm{diag}(y)
\end{aligned}
$$

This is Lagrange dual function. Dual formulation is OK since quadratic function with linear constraints fulfills **Slater's** conditions.

OSH dual formulation, cont'd

Step 4 Identify the QP components

$$\left\{\begin{array}{l} \quad \arg\max_\alpha \alpha \mathbf{1} - \frac{1}{2}\alpha^T YXX^T Y\alpha \\ \text{such that} \\ \quad \alpha_i \geq 0 \quad \forall i \\ \quad \sum \alpha_i y_i = 0 \end{array}\right.$$

The general form of a QP problem is

$$\left\{\begin{array}{l} \quad \arg min_\alpha \alpha^T Q\alpha + c^T\alpha \\ \text{such that} \\ \quad A\alpha \leq b \\ \quad E\alpha = d \end{array}\right.$$

- Identify Q, c, A, b, E and d?
- How do we get β?

Two more things...

How do we find β_0?

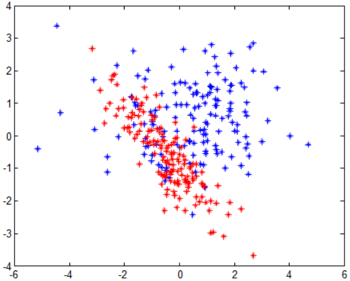
- For the support points we have $y_i(x_i\beta + \beta_0) = 1$
- Use one of the support points to calculate β_0

How do we predict class belonging?

- Along the decision line we have $x\beta + \beta_0 = 0$
- Along the support lines we have $x\beta + \beta_0 = \pm 1$
- Decision based on the side of the line
  $\hat{y}(x_{\text{new}}) = \text{sign}(x_{\text{new}}\beta + \beta_0)$

Wait a minute... Margin??

- For overlapping data, there is no solution
  - What's the use?



- Can be modified into the **Support Vector Machine**
  - Handles overlapping observations.
  - Kernel trick for non-linear data.

Support Vector Machine

- Most classification problems have overlapping classes.
- Let us modify the OSH such that we allow for some overlap
- This is the **Support Vector Machine**
- Used together with the kernel trick SVM is one of our most flexible classifiers

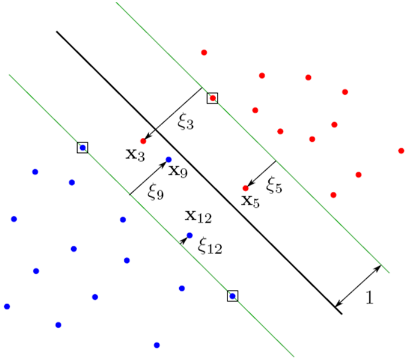
SVM Cost Function

We got OSH from

$$\begin{cases} \quad \arg\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 \\ \text{such that} \\ \quad y_i(x_i\beta - \beta_0) \geq 1 \quad \forall i \end{cases}$$

Now, allow some overlap

$$\begin{cases} \quad \arg\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + \lambda \sum_{i=1}^{n} \xi_i \\ \text{such that} \\ \quad y_i(x_i\beta - \beta_0) \geq 1 - \xi_i \quad \forall i \\ \quad \xi_i \geq 0 \quad \forall i \end{cases}$$



We give our self a **budget for overlap**.

Smaller budget - larger λ - noisier solution

Solving the SVM Problem

Similar to OSH

$$\begin{cases} \quad \arg\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + \lambda \sum_{i=1}^{n} \xi_i \\ \text{such that} \\ \quad y_i(x_i\beta + \beta_0) \geq 1 - \xi_i \quad \forall i \\ \quad \xi_i \geq 0 \quad \forall i \end{cases}$$

Lagrange multiplier, differentiate, plug back...

$$\begin{cases} \quad \arg\max_{\alpha} \alpha\mathbf{1} - \frac{1}{2}\alpha^T YXX^T Y\alpha \\ \text{such that} \\ \quad \sum \alpha_i y_i = 0 \\ \quad 0 \leq \alpha_i \leq \lambda \quad \forall i \end{cases}$$

Comparison with OSH

Optimal separating hyperplanes

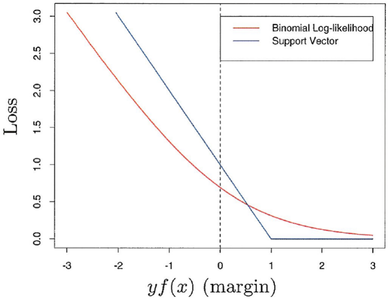
$$\begin{cases} \quad \arg\max_\alpha \alpha\mathbf{1} - \frac{1}{2}\alpha^T YXX^T Y\alpha \\ \text{such that} \\ \quad \alpha_i \geq 0 \quad \forall i \\ \quad \sum \alpha_i y_i = 0 \end{cases}$$

Support vector machine

$$\begin{cases} \quad \arg\max_\alpha \alpha\mathbf{1} - \frac{1}{2}\alpha^T YXX^T Y\alpha \\ \text{such that} \\ \quad 0 \leq \alpha_i \leq \lambda \quad \forall i \\ \quad \sum \alpha_i y_i = 0 \end{cases}$$

Both are quadratic programming problems with linear constraints

Comparison with logistic regression



With $f(x) = x\beta + \beta_0$ and $y_i \in \{-1, 1\}$, consider

$$\min_{\beta,\beta_0} \sum_{i=1}^{N} (1 - y_i f(x_i))_+ + \frac{\lambda}{2}||\beta||^2$$

This hinge loss criterion is equivalent to the SVM. Compare with

$$\min_{\beta,\beta_0} \sum_{i=1}^{N} \log(1 + e^{-y_i f(x_i)}) + \frac{\lambda}{2}||\beta||^2$$

(In Lecture 3 we used $y_i \in \{0, 1\}$.)

This is the ML formulation of **ridged logistic regression**

Basis expansion and kernels

- We can do SVM (and OSH) on a transformed feature space
- Transformed features gives non-linear decision boundaries.
- With **the Kernel trick** we can use an infinite dimensional feature expansion

Non-linear SVM

Let's try basis expansions!

$$\begin{cases} \quad \arg\max_\alpha \alpha\mathbf{1} - \frac{1}{2}\alpha^T YXX^T Y\alpha \\ \text{such that} \\ \quad 0 \le \alpha_i \le \lambda \quad \forall i \\ \quad \sum \alpha_i y_i = 0 \end{cases}$$

Use $h(X)$ instead of X,

$$\begin{cases} \quad \arg\max_\alpha \alpha\mathbf{1} - \frac{1}{2}\alpha^T Yh(X)h(X)^T Y\alpha \\ \text{such that} \\ \quad 0 \le \alpha_i \le \lambda \quad \forall i \\ \quad \sum \alpha_i y_i = 0 \end{cases}$$

- $h(X): \; R^p \to R^M$, *e.g.* $[x_1 \;\; x_2] \to [x_1 \;\; x_2^2 \;\; x_1 x_2]$
- $h(X)h(X)^T$ is of size $n \times n$

The kernel trick

The term $h(X)h(X)^T$ does not depend on M, the number of basis functions.

We only need to specify $K(X)$ such that $h(X)h(X)^T = K(X)$ - we call K a **kernel**. Then h is implicitly defined by K.

Common kernels

Polynomial $K_{i,j} = (1 + x_i x_j^T)^d$ (x_i is observation i, ie row i in X)

Radial $K_{i,j} = \exp\left(-\frac{1}{c}||x_i - x_j||^2\right)$

Gaussian $K_{i,j} = \exp\left(-\frac{1}{2\sigma^2}||x_i - x_j||^2\right)$

Neural network $K_{i,j} = \tanh(c_1 x_i x_j^T + c_2)$

SVM with kernels

The new optimization problem

$$\begin{cases} \quad \arg\max_\alpha \alpha\mathbf{1} - \frac{1}{2}\alpha^T YKY\alpha \\ \text{such that} \\ \quad 0 \le \alpha_i \le \lambda \quad \forall i \\ \quad \sum \alpha_i y_i = 0 \end{cases}$$

To classify a new observations

$$\begin{aligned} \hat{y}_{\text{new}} &= \text{sign}(\beta h(x_{\text{new}}) + \beta_0) \\ &= \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_{\text{new}}, x_i) + b_0\right) \end{aligned}$$

Calculate b_0 using one of the points, i, on the margin,

$$b_0 = y_i - \sum_{j=1}^{n} \alpha_j y_j K(x_i, x_j)$$

Phew!

- We have found an efficient way of maximizing the margin between classes
- Of course, there are software packages for you!

SVM in Matlab

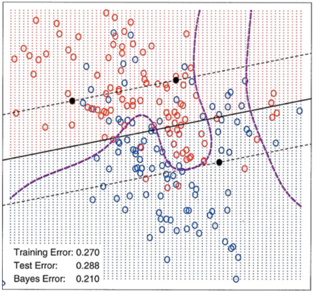
```
SVMModel = fitcsvm(X,Y,Name,Value);
Yhat = predict(SVMModel,X);
```

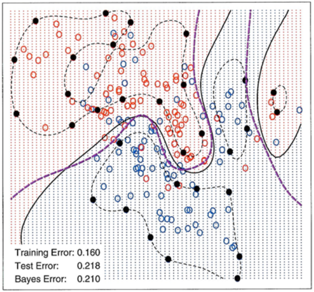
Name	Value
`'BoxConstraint'`	λ (`Inf` for OSH.)
`'KernelFunction'`	`'linear'`, `'rbf'`, `'polynomial'`
`'KernelScale'`	*c*
`'PolynomialOrder'`	positive integer
`'Standardize'`	good idea for SVM also

Use `'OptimizeHyperparameters'` to select parameters for tuning and `'HyperparameterOptimizationOptions'` to define a grid-search and cross validation.

Example

Linear SVM and enlarged feature space using RBF kernel





Model selection and SVM

Use SVM together with Radial Basis Function kernel

- $K_{i,j} = \exp\left(-\frac{1}{c}||x_i - x_j||^2\right)$
- This give one parameter c

From the SVM loss function

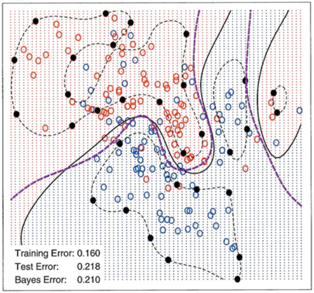
- $\arg\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + \lambda \sum_{i=1}^{n} \xi_i$
- This gives another parameter λ

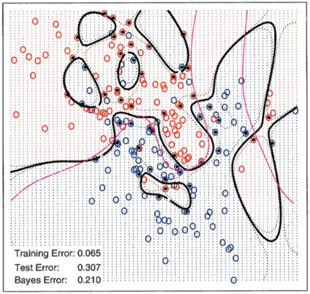
Select parameters using cross validation

- Extensive search for c
- λ less crucial, try different values

Overfitting

Overfitting is easy in an enlarged feature space!





Caveats

- Kernel methods do not scale well. Limited to around 10000-20000 observations
- Kernel methods do not do variable selection in any reasonable or automatic way
  - With more features than observations there is always a separating hyperplane
  - Actually infinitely many which we have to choose between
- Potential problem with large number of features if many of them are garbage
- SVM do not generalize gracefully when the number of classes are more than two
  - Frequently used for multiclass classification anyway

Summary

- Constrained optimization
  - Lagrange multipliers
  - Primal and dual formulation
- Optimal separating hyperplanes
  - Margin
  - Support vectors and support points
- Support vector machine
  - Budget for overlap
  - Kernel trick

Questions?