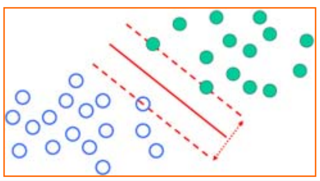
Machine Learning

10-701/15-781, Fall 2011

Alternative Strategies of Learning (2)

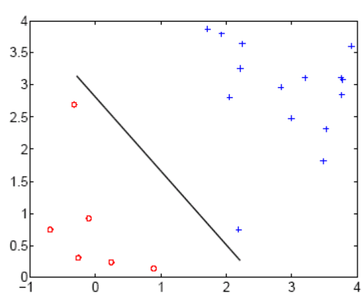
Support Vector Machines

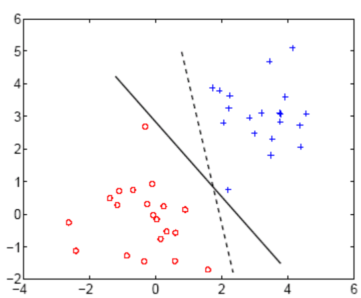


Eric Xing

Lecture 17, November 9, 2011

What is a good Decision Boundary?



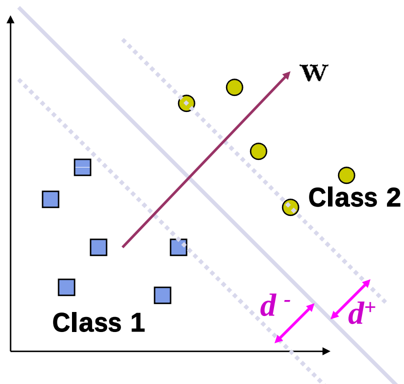


- Why we may have such boundaries?
  - Irregular distribution
  - Imbalanced training sizes
  - outliners

Classification and Margin

- Parameterzing decision boundary
  - Let w denote a vector orthogonal to the decision boundary, and b denote a scalar "offset" term, then we can write the decision boundary as:

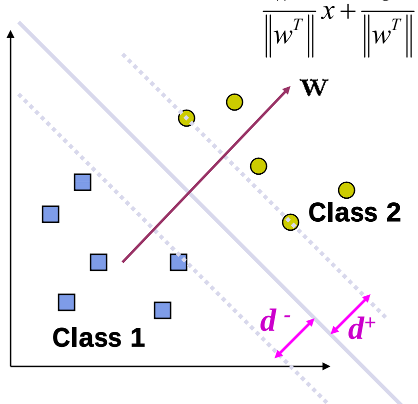
$$w^T x + b = 0$$



Classification and Margin

- Parameterzing decision boundary
  - Let w denote a vector orthogonal to the decision boundary, and b denote a scalar "offset" term, then we can write the decision boundary as:

$$\frac{w^T}{\|w^T\|} x + \frac{b}{\|w^T\|} = 0$$



- **Margin**

$(w^T x_i + b)/\|w\| > +c/\|w\|$ for all x_i in class 2

$(w^T x_i + b)/\|w\| < -c/\|w\|$ for all x_i in class 1

Or more compactly:

$(w^T x_i + b) y_i / \|w\| > c/\|w\|$

The margin between two points

$m = d^- + d^+ =$

Maximum Margin Classification

- The margin is:

$$m = \frac{w^T}{\|w\|}\left(x_{i^*} - x_{j^*}\right) = \frac{2c}{\|w\|}$$

- Here is our Maximum Margin Classification problem:

$$\begin{aligned} \max_w \quad & \frac{2c}{\|w\|} \\ \text{s.t} \quad & y_i(w^T x_i + b)/\|w\| \geq c/\|w\|, \quad \forall i \end{aligned}$$

Maximum Margin Classification, con'd.

- The optimization problem:

$$\begin{aligned} \max_{w,b} \quad & \frac{c}{\|w\|} \\ \text{s.t} \quad & y_i(w^T x_i + b)/\|w\| \geq c/\|w\|, \quad \forall i \end{aligned}$$

- But note that the magnitude of c merely scales w and b, and does not change the classification boundary at all! (why?)
- So we instead work on this cleaner problem:

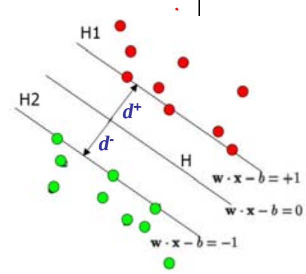
$$\begin{aligned} \max_{w,b} \quad & \frac{1}{\|w\|} \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq 1, \quad \forall i \end{aligned}$$

- The solution to this leads to the famous **Support Vector Machines** --- believed by many to be the best "off-the-shelf" supervised learning algorithm

Support vector machine

- A convex quadratic programming problem with linear constrains:

$$\max_{w,b} \quad \frac{1}{\|w\|}$$
$$\text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$



  - The attained margin is now given by $\frac{1}{\|w\|}$
  - Only a few of the classification constraints are relevant ➔ **support vectors**

- Constrained optimization
  - We can directly solve this using commercial quadratic programming (QP) code
  - But we want to take a more careful investigation of Lagrange duality, and the solution of the above in its dual form.

➔ deeper insight: support vectors, kernels …

➔ more efficient algorithm

Digression to Lagrangian Duality

- The Primal Problem

Primal:

$$\min_w \quad f(w)$$
$$\text{s.t.} \quad g_i(w) \leq 0, \quad i = 1, \ldots, k$$
$$h_i(w) = 0, \quad i = 1, \ldots, l$$

The generalized Lagrangian:

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

the α's ($\alpha_i \geq 0$) and βs are called the Lagarangian multipliers

Lemma:

$$\max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{o/w} \end{cases}$$

A re-written Primal:

$$\min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

Lagrangian Duality, cont.

- Recall the Primal Problem:

$$\min_w \max_{\alpha,\beta,\alpha_i \geq 0} \mathcal{L}(w,\alpha,\beta)$$

- The Dual Problem:

$$\max_{\alpha,\beta,\alpha_i \geq 0} \min_w \mathcal{L}(w,\alpha,\beta)$$

- **Theorem (weak duality):**

$$d^* = \max_{\alpha,\beta,\alpha_i \geq 0} \min_w \mathcal{L}(w,\alpha,\beta) \leq \min_w \max_{\alpha,\beta,\alpha_i \geq 0} \mathcal{L}(w,\alpha,\beta) = p^*$$

- **Theorem (strong duality):**

  Iff there exist a saddle point of $\mathcal{L}(w,\alpha,\beta)$, we have

$$d^* = p^*$$

The KKT conditions

- If there exists some saddle point of $\mathcal{L}$, then the saddle point satisfies the following "Karush-Kuhn-Tucker" (KKT) conditions:

$$\frac{\partial}{\partial w_i}\mathcal{L}(w,\alpha,\beta) = 0, \quad i = 1,\ldots,k$$

$$\frac{\partial}{\partial \beta_i}\mathcal{L}(w,\alpha,\beta) = 0, \quad i = 1,\ldots,l$$

$$\alpha_i g_i(w) = 0, \quad i = 1,\ldots,m$$

$$g_i(w) \leq 0, \quad i = 1,\ldots,m$$

$$\alpha_i \geq 0, \quad i = 1,\ldots,m$$

- **Theorem**: If w^*, α^* and β^* satisfy the KKT condition, then it is also a solution to the primal and the dual problems.

Solving optimal margin classifier

- Recall our opt problem:

$$\max_{w,b} \quad \frac{1}{\|w\|}$$
$$\text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$

- This is equivalent to

$$\min_{w,b} \quad \frac{1}{2} w^T w$$
$$\text{s.t} \quad 1 - y_i(w^T x_i + b) \leq 0, \quad \forall i \qquad (*)$$

- Write the Lagrangian:

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i \left[y_i (w^T x_i + b) - 1 \right]$$

  - Recall that (*) can be reformulated as $\min_{w,b} \max_{\alpha_i \geq 0} \mathcal{L}(w,b,\alpha)$
    Now we solve its **dual problem**: $\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha)$

The Dual Problem

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i \left[y_i (w^T x_i + b) - 1 \right]$$

$$\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha)$$

- We minimize $\mathcal{L}$ with respect to w and b first:

$$\nabla_w \mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0, \qquad (*)$$

$$\nabla_b \mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i y_i = 0, \qquad (**)$$

Note that **(*)** implies: $w = \sum_{i=1}^{m} \alpha_i y_i x_i \qquad$ **(***)**

- Plug (***) back to $\mathcal{L}$, and using (**), we have:

$$\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

The Dual problem, cont.

- Now we have the following dual opt problem:

$$\max_\alpha \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, k$$

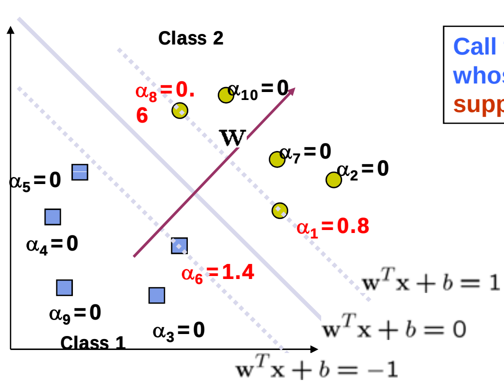
$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- This is, (again,) a **quadratic programming** problem.
  - A global maximum of α_i can always be found.
  - But what's the big deal??
  - Note two things:
    1. w can be recovered by $w = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ **See next ...**
    2. The "kernel" $\mathbf{x}_i^T \mathbf{x}_j$ **More later ...**

I. Support vectors

- Note the KKT condition --- only a few α_i's can be nonzero!!

$$\alpha_i g_i(w) = 0, \quad i = 1, \ldots, m$$



Call the training data points whose α_i's are nonzero the support vectors (SV)

Support vector machines

- Once we have the Lagrange multipliers $\{\alpha_i\}$, we can reconstruct the parameter vector w as a weighted combination of the training examples:

$$w = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

- For testing with a new data z
  - Compute $$w^T z + b = \sum_{i \in SV} \alpha_i y_i \left(\mathbf{x}_i^T z \right) + b$$

    and classify z as class 1 if the sum is positive, and class 2 otherwise
  - Note: w need not be formed explicitly

Interpretation of support vector machines

- The optimal w is a linear combination of a small number of data points. This "sparse" representation can be viewed as data compression as in the construction of kNN classifier

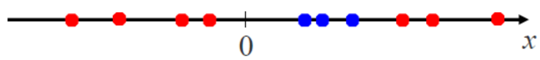
- To compute the weights $\{\alpha_i\}$, and to use support vector machines we need to specify only the inner products (or kernel) between the examples $\mathbf{x}_i^T \mathbf{x}_j$

- We make decisions by comparing each new example z with only the support vectors:

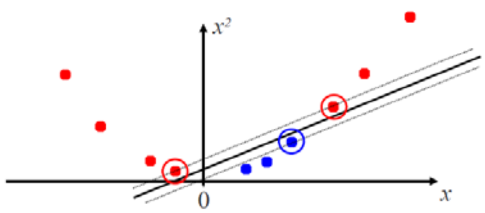
$$y^* = \operatorname{sign}\left(\sum_{i \in SV} \alpha_i y_i \left(\mathbf{x}_i^T z \right) + b \right)$$

II. The Kernel Trick

- Is this data linearly-separable?



- How about a quadratic mapping $\phi(x_i)$?



II. The Kernel Trick

- Recall the SVM optimization problem

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function K by $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

II. The Kernel Trick

- Computation depends on feature space
  - Bad if its dimension is much larger than input space

$$\max_\alpha \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

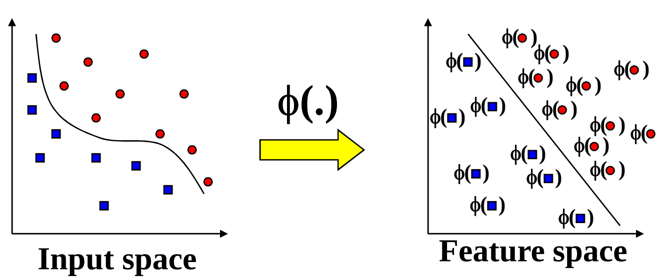
$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

Where $K(x_i,x_j) = \phi(x_i)^t\, \phi(x_j)$

$$y^*(z) = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, z) + b\right)$$

Transforming the Data



Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

An Example for feature mapping and kernels

- Consider an input $\mathbf{x}=[x_1,x_2]$
- Suppose $\phi(.)$ is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2$$

- An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right) \right\rangle =$$

- So, if we define the **kernel function** as follows, there is no need to carry out $\phi(.)$ explicitly

$$K(\mathbf{x},\mathbf{x}') = \left(1+\mathbf{x}^T\mathbf{x}'\right)^2$$

More examples of kernel functions

- Linear kernel (we've seen it)

$$K(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\mathbf{x}'$$

- Polynomial kernel (we just saw an example)

$$K(\mathbf{x},\mathbf{x}') = \left(1+\mathbf{x}^T\mathbf{x}'\right)^p$$

where p = 2, 3, … To get the feature vectors we concatenate all pth order polynomial terms of the components of x (weighted appropriately)

- Radial basis kernel

$$K(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\left\|\mathbf{x}-\mathbf{x}'\right\|^2\right)$$

In this case the feature space consists of functions and results in a non-parametric classifier.

The essence of kernel

- Feature mapping, but "without paying a cost"
  - E.g., polynomial kernel
    $$K(x, z) = (x^T z + c)^d$$
  - How many dimensions we've got in the new space?
  - How many operations it takes to compute K()?

- Kernel design, any principle?
  - K(x,z) can be thought of as a similarity function between x and z
  - This intuition can be well reflected in the following "Gaussian" function
    (Similarly one can easily come up with other K() in the same spirit)
    $$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$
  - Is this necessarily lead to a "legal" kernel?
    (in the above particular case, K() is a legal one, do you know how many dimension ϕ(x) is?

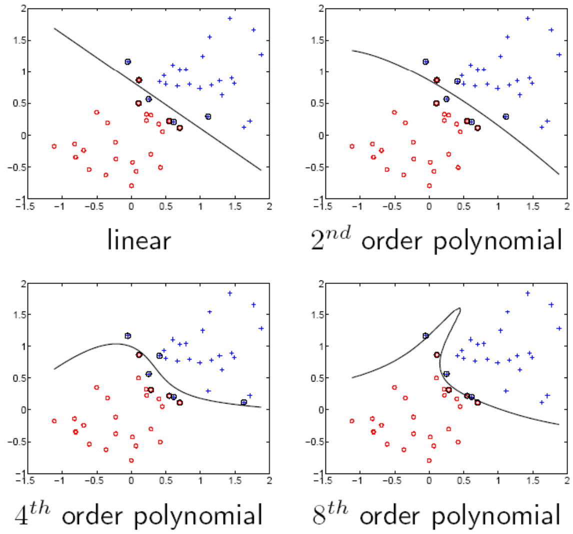
Kernel matrix

- Suppose for now that K is indeed a valid kernel corresponding to some feature mapping ϕ, then for $x_1, \ldots, x_m$, we can compute an $m\times m$ matrix $K = \{K_{i,j}\}$, where $K_{i,j} = \phi(x_i)^T\phi(x_j)$

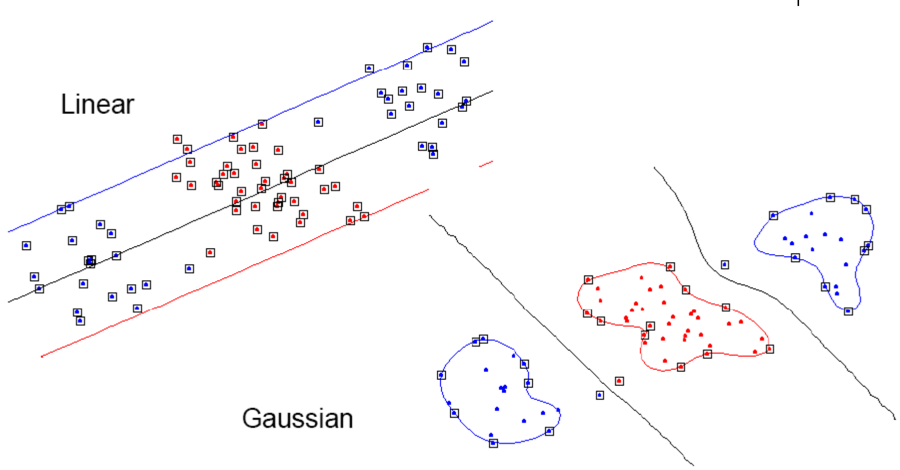
- This is called a kernel matrix!

- Now, if a kernel function is indeed a valid kernel, and its elements are dot-product in the transformed feature space, it must satisfy:
  - Symmetry $K=K^T$
    proof $K_{i,j} = \phi(x_i)^T\phi(x_j) = \phi(x_j)^T\phi(x_i) = K_{j,i}$
  - Positive –semidefinite $y^T K y \geq 0 \quad \forall y$
    proof?
  - Mercer's theorem

SVM examples



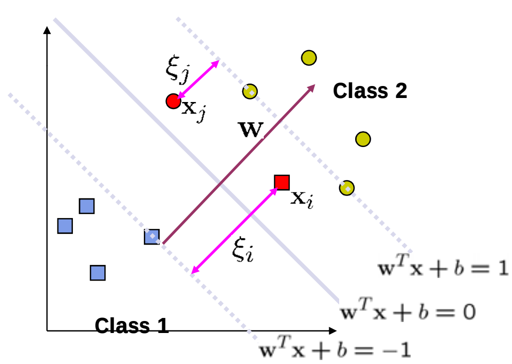
Examples for Non Linear SVMs – Gaussian Kernel



Example Kernel

- x_i is a bag of words
- Define $\phi(x_i)$ as a count of every n-gram up to n=k in x_i.
  - This is huge space 26^k
  - What are we measuring by $\phi(x_i)^t\ \phi(x_j)$?
- Can we compute the same quantity on input space?
  - Efficient linear dynamic program!
- Kernel is a measure of similarity
- Must be positive semi-definite

Non-linearly Separable Problems



- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $w^T x+b$
- ξ_i approximates the number of misclassified samples

Soft Margin Hyperplane

- Now we have a slightly different opt problem:

$$\min_{w,b} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i$$

$$\text{s.t} \quad \begin{array}{l} y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \forall i \\ \xi_i \geq 0, \quad \forall i \end{array}$$

  - ξ_i are "slack variables" in optimization
  - Note that ξ_i=0 if there is no error for $\mathbf{x}_i$
  - ξ_i is an upper bound of the number of errors
  - *C* : tradeoff parameter between error and margin

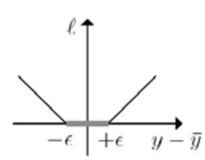
Hinge Loss

- Remember Ridge regression
  - Min [squared loss + λ $w^t w$]

- How about SVM?

$$\text{argmin}_{\{w,b\}} \underbrace{w^t w}_{\text{regularization}} + \underbrace{\lambda \sum_{1}^{m} \max(1 - y_i(w^t x_i + b), 0)}_{\text{Loss: hinge loss}}$$

regularization **Loss: hinge loss**



The Optimization Problem

- The dual of this new constrained optimization problem is

$$\max_\alpha \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on α_i now
- Once again, a QP solver can be used to find α_i

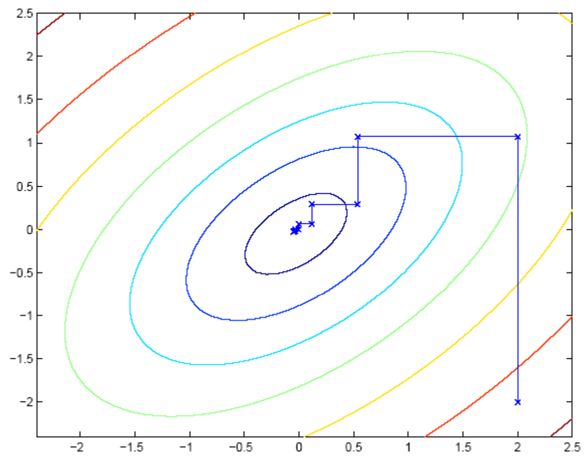
The SMO algorithm

- Consider solving the unconstrained opt problem:

$$\max_\alpha W(\alpha_1, \alpha_2, \ldots, \alpha_m)$$

- We've already seen several opt algorithms!
  - ?
  - ?
  - ?
- Coordinate ascend:

Coordinate ascend



Sequential minimal optimization

- Constrained optimization:

$$\max_\alpha \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- Question: can we do coordinate along one direction at a time (i.e., hold all $\alpha_{[-i]}$ fixed, and update α_i?)

The SMO algorithm

Repeat till convergence

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).

2. Re-optimize J(α) with respect to α_i and α_j, while holding all the other α_k 's ($k \neq i; j$) fixed.

Will this procedure converge?

Convergence of SMO

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

KKT:

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, k$$

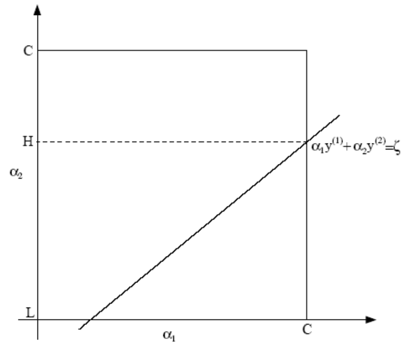
$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- Let's hold α_3 ,…, α_m fixed and reopt J w.r.t. α_1 and α_2

Convergence of SMO

- The constraints:

$$\alpha_1 y_1 + \alpha_2 y_2 = \xi$$
$$0 \leq \alpha_1 \leq C$$
$$0 \leq \alpha_2 \leq C$$



- The objective:

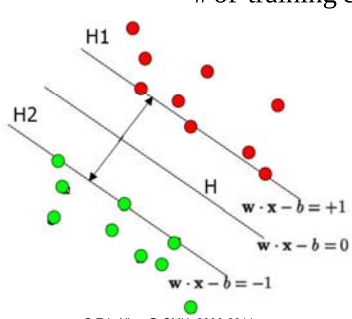
$$\mathcal{J}(\alpha_1, \alpha_2, \ldots, \alpha_m) = \mathcal{J}((\xi - \alpha_2 y_2) y_1, \alpha_2, \ldots, \alpha_m)$$

- Constrained opt:

Cross-validation error of SVM

- The leave-one-out cross-validation error does not depend on the dimensionality of the feature space but only on the # of support vectors!

$$\text{Leave-one-out CV error} = \frac{\#\text{ support vectors}}{\#\text{ of training examples}}$$



Summary

- Max-margin decision boundary
- Constrained convex optimization
  - Duality
  - The KTT conditions and the support vectors
  - Non-separable case and slack variables
  - The SMO algorithm