ECE 2372 - Pattern Recognition

Lecture 9

Recap: The "kernel trick":

- Many ML algorithms only involve the data through inner products
- For many interesting feature maps ϕ, the function

  $$k(x, x') := \langle \phi(x), \phi'(x) \rangle$$

  has a simple, closed form expression that can be evaluated **without explicitly calculating** $\phi(x)$ and $\phi'(x)$.

The "kernel trick" in classification It is possible to "kernelize":

- LDA
- Logistic regression
- Perceptron learning algorithms
- Maximum margin hyperplanes
  - **support vector machines**

$$\begin{aligned} min_{\mathrm{w},b,\xi} \; \frac{1}{2}\|\mathrm{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ \text{s.t. } y_i(\mathrm{w}^T x_i + b) \geq 1 - \xi_i, \qquad i = 1, ..., n \\ \xi_i \geq \qquad i = 1, ..., n \end{aligned}$$

We need (a detour) to spend a bit of time learning about constrained optimization, in order to understand both

- how to kernelize the maximum margin optimization problem
- how to actually solve this optimization problem with a practical algorithm

Constrained Optimization

A general constrained optimization problem has the form

$$\begin{aligned} &\min_x \ f(x) \\ &\quad \text{s.t. } g_i(x) \leq 0 \quad i = 1, ..., m \\ &\quad \text{s.t. } h_i(x) = 0 \quad i = 1, ..., p \end{aligned}$$

where $x \in \mathbb{R}^d$. We call $f(x)$ the **objective function**. The direction of inequality constrain is arbitrary but it is useful to stick with a convention. You can always convert inequalities to look this standard form.

If x satisfies all the constraints, we say that x is **feasible**. We will assume that $f(x)$ is defined for all feasible x.

Lagrangian duality

By considering the **dual**, we can **bound** or **solve** the original optimization problem via a different optimization problem. (In the case of SVM it will help us to find a different way of solving the problem)

The Lagrangian function:

$$L(\mathrm{x}, \lambda, \nu) := f(\mathrm{x}) + \sum_{i=1}^{m} \lambda_i \ g_i(\mathrm{x}) + \sum_{i=1}^{p} \nu_i \ h_i(\mathrm{x})$$

$\lambda = [\lambda_1, ..., \lambda_m]^T$ and $\nu = [\nu_1, ..., \nu_p]$ are called **Lagrange multipliers** or **dual variables**

(Lagrange) dual function

$$L_D(\lambda, \nu) = \min_{\mathrm{X}} \ L(\mathrm{x}, \lambda, \nu)$$

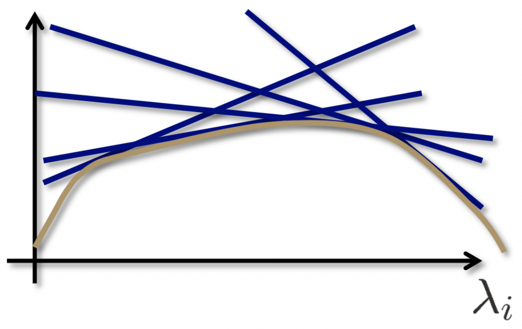
The dual optimization problem

We can then define the dual problem as

$$\begin{aligned} &\max_{\lambda,\nu} \ L_D(\lambda, \nu) \\ &\quad \text{s.t. } \lambda_i \geq 0 \quad i = 1, ..., m \end{aligned}$$

Why do we constrain $\lambda_i \geq 0$?

Note that no matter the choice of f, g_i or h_i, $L_D(\lambda, \nu)$ is always **concave**. It is the point-wise minimum of a family of affine functions.



The primal optimization problem

The primal function is

$$L_P(\mathrm{x}) := \max_{\lambda,\nu:\lambda_i\geq 0}\ L(\mathrm{x},\lambda,\nu)$$

and the primal optimization problem is

$$\min_{\mathrm{x}}\ L_P(\mathrm{x}) := \min_{\mathrm{x}}\ \max_{\lambda,\nu:\lambda_i\geq 0}\ L(\mathrm{x},\lambda,\nu)$$

Contrast this with the dual optimization problem

$$\max_{\lambda,\nu:\lambda_i\geq 0}\ \min_{\mathrm{x}}\ L(\mathrm{x},\lambda,\nu)$$

- Why is it called the primal?

  $$\min_{\mathrm{x}}\ L_P(\mathrm{x}) := \min_{\mathrm{x}}\ \max_{\lambda,\nu:\lambda_i\geq 0}\ L(\mathrm{x},\lambda,\nu)$$

  Suppose x is feasible

  $$\begin{aligned} L_P(\mathrm{x}) &= \max_{\lambda,\nu:\lambda_i\geq 0}\ f(x) + \sum_{i=1}^{m} \lambda_i\ g_i(\mathrm{x}) + \sum_{i=1}^{p} \nu_i\ h_i(\mathrm{x}) \\ &= f(x) \end{aligned}$$

  Suppose x is not feasible

  $$L_P(\mathrm{x}) = \infty$$

  so that means if you are trying to minimize this $L_P(\mathrm{x})$, you will definitely pick something always feasible. And if it is feasible, what you are minimizing is just the original objective function.

  The primal encompasses the original problem (i.e., has the same solution), yet it is **unconstrained**. This is kind of convenient.

> Also, flashing forward, this is the primal problem: $\min_{\mathrm{X}} \max_{\lambda,\nu:\lambda_i\geq 0} L(\mathrm{x},\lambda,\nu)$, and we also have the dual problem where we switch the order of min-max, we first minimize w.r.t. x than we maximize w.r.t. dual variables. So under certain conditions, it doesn't matter which order you do, we are gonna end up seeing especially like with SVMs, sometimes it is a lot easier to solve the problem by forming this dual function and maximizing over the λ's...And you are gonna able to go back and forth between these two formulations (under certain conditions). So if you can solve one, you can generate the optimal solution to the other, sometimes there are basically the same problem...That's kind of where we are headed...

Weak duality

$$\begin{aligned} d^* &:= \max_{\lambda,\nu:\lambda_i\geq 0} \min_{\mathrm{X}} L(x,\lambda,\nu) \\ &\leq \min_{\mathrm{X}} \max_{\lambda,\nu:\lambda_i\geq 0} L(x,\lambda,\nu) := p^* \end{aligned}$$

To show this, let $\tilde{x}$ be feasible. Then for any λ,ν with $\lambda_i \geq 0$,

$$L(\tilde{x},\lambda,\nu) = f(\tilde{x}) + \sum_i \lambda_i g_i(\tilde{x}) + \sum_i \nu_i h_i(\tilde{x})$$

(So that means that when we look at the dual function where we taking the Lagrangian and choosing the x that minimizes that–every possible x. This is always gonna be smaller than just minimizing over some subset of x's–so we can always say this is smaller than minimizing over "just the feasible x".) Hence

$$\begin{aligned} L_D(\lambda,\nu) &= \min_x L(\mathrm{x},\lambda,\nu) \\ &\leq min_{\text{feasible } \tilde{x}} \ L(\tilde{x},\lambda,\nu) \\ &\leq min_{\text{feasible } \tilde{x}} \ f(\tilde{x}) = p^* \end{aligned}$$

Thus, for any λ,ν with $\lambda_i \geq 0$, we have

$$\min_{\mathrm{X}} L(x,\lambda,\nu) \leq p^*$$

Since this holds for any λ,ν , it is also true if we take the max of λ,ν:

$$d^* = \max_{\lambda,\nu:\lambda_i\geq 0} \min_{\mathrm{X}} L(x,\lambda,\nu) \leq p^*$$

Why is this useful? This is actually useful mostly in algorithmic sense. Say we had a Primal optimization problem where the function we are trying to minimize is non-convex. Then it is hard to know, you actually found the optimum. Because, you might find the spot where the gradient is 0, but there are lots of local minima, so you don't necessarily know how close to the optimum you are... But what you can sometimes do : you can form this dual-problem (which I said always concave)–so you are maximizing a concave function, so that means there is only one unique maximum value of that (global).

Duality Gap

For any optimization problem, we can solve the dual (which is always a concave function problem), and obtain a lower bound d^* on p^* (the optimal value of the objective function for the original problem).

In general all we know if that p^* is at least as big as the d^*.

The difference $p^* - d^*$ is called the **"duality gap"**. In general, all we can say about the duality gap is that $p^* - d^* \geq 0$, bu in a lot of actual problems, we know a lot more than this...which is called strong duality

Strong duality

If $p^* = d^*$, we say that strong duality holds.

Theorem

If the original problem is **convex**, meaning that f and the g_i are convex and the h_i are affine, then under certain **constraint qualifications**, $p^* = d^*$.

Example constraint qualifications:

- All the g_i are affine
- Strict feasibility: There exists an x such that $h_i(\mathrm{x}) = 0$ for all i and $g_i(\mathrm{x}) < 0$ for all i.

Last thing we need for SVM is KKT conditions: what are they gonna do for us is basically they allow us to really understand when we have this strong duality, and how do we go back and forth between the primal and dual optimization problems.

KKT (Karush-Kuhn-Tucker) conditions

Assume that f, g, h are all differentiable.

The Karush-Kuhn-Tucker (KKT) conditions are a set of properties that must hold under strong duality.

The KKT conditions will allow us to translate between solutions of the primal and dual optimization problems.

$$x^* = \arg\min_{x} \; L_P(\mathrm{x})$$

$$\updownarrow$$

$$(\lambda^*, \nu^*) = \arg\max_{\lambda, \nu : \lambda_i \geq 0} L_D(\lambda, \nu)$$

1. $\nabla f(\mathrm{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(\mathrm{x}^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(\mathrm{x}^*) = 0$

2. $g_i(\mathrm{x}^*) \leq 0, \quad i = 1, ..., m$

3. $h_i(\mathrm{x}^*) = 0, \quad i = 1, ..., p$

4. $\lambda_i^* \geq 0, \quad i = 1, ..., m$

5. $\lambda_i^* g_i(\mathrm{x}^*) = 0$, $i = 1, ..., m$ **(complementary slackness)**

KKT conditions: Necessity:

Theorem:

If $p^* = d^*$, x^* is primal optimal, and (λ^*, ν^*) is dual optimal then the KKT conditions hold.

Proof:

- 2 and 3 hold because x^* must be feasible.
- 4 hold by the definition of the dual problem
- To prove 5, note that from strong duality, we have

$$\begin{aligned} f(\mathrm{x}^*) &= L_D(\lambda^*, \nu^*) \\ &= \min\nolimits_{\mathrm{X}} \ f(\mathrm{x}) + \sum_{i=1}^{m} \lambda_i^* g_i(\mathrm{x}) + \sum_{i=1}^{p} \nu_i^* h_i(\mathrm{x}) \\ &\leq f(\mathrm{x}^*) + \sum_{i=1}^{m} \lambda_i^* g_i(\mathrm{x}^*) + \sum_{i=1}^{p} \nu_i^* h_i(\mathrm{x}^*) \\ &\leq f(\mathrm{x}^*) \quad \text{by 2,3,4} \end{aligned}$$

  - This shows that the two inequalities are actually equalities, and hence the equality of the last two lines implies

$$f(\mathrm{x}^*) + \sum_{i=1}^{m} \lambda_i^* g_i(\mathrm{x}^*) + \sum_{i=1}^{p} \nu_i^* h_i(\mathrm{x}^*) = f(\mathrm{x}^*)$$

    which implies that $\sum_{i=1}^{m} \lambda_i^* g_i(\mathrm{x}^*) = 0$, which is 5 (complementary slackness).
  - Equality of 2^{nd} and 3^{rd} lines implies that x^* is a minimizer of $L(\mathrm{x}, \lambda^*, \nu^*)$ with respect to x, and hence

$$\nabla L(\mathrm{x}^*, \lambda^*, \nu^*) = 0$$

    which establishes 1.

KKT conditions: Sufficiency

Theorem

If the original problem is convex, meaning that f and the g_i are convex and the h_i are affine, and $(\tilde{x}, \tilde{\lambda}, \tilde{\nu})$ satisfy the KKT conditions, then $\tilde{x}$ is a primal optimal $(\tilde{\lambda}, \tilde{\nu})$ is dual optimal, and the duality gap is zero.

Proof

- 2 and 3 imply that $\tilde{x}$ is feasible
- 4 implies that $L(x, \tilde{\lambda}, \tilde{\nu})$ is convex in x, and so 1 means that $\tilde{x}$ is a minimizer of $L(x, \tilde{\lambda}, \tilde{\nu})$.
- Thus

$$\begin{aligned} L_D(\tilde{\lambda}, \tilde{\nu}) &= L(\tilde{x}, \tilde{\lambda}, \tilde{\nu}) \\ &= f(\tilde{\mathrm{x}}) + \sum_{i=1}^{m} \tilde{\lambda}_i^* g_i((\tilde{\mathrm{x}}) + \sum_{i=1}^{p} \tilde{\nu}_i^* h_i((\tilde{\mathrm{x}}) \\ &= f(\tilde{x}) \text{ by feasibility and 5} \end{aligned}$$

  - If $L_D(\tilde{\lambda}, \tilde{\nu}) = f(\tilde{\mathrm{x}})$ then

$$\begin{aligned} d^* &= max_{\lambda,\nu:\lambda_i \geq 0}\ L_D(\lambda, \nu) \\ &\geq L_D(\tilde{\lambda}, \tilde{\nu}) \\ &= f(\tilde{\mathrm{x}}) \\ &\geq p^* \end{aligned}$$

  - But earlier we proved that $d^* \leq p^*$, and hence it must actually be that $d^* = p^*$, and thus we have
    - zero duality gap
    - $\tilde{x}$ is primal optimal
    - $(\tilde{\lambda}, \tilde{\nu})$ is dual optimal

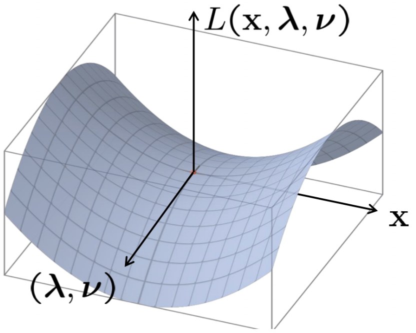
KKT conditions: The bottom line

If a constrained optimization problem is

- differentiable
- convex

then the KKT conditions are necessary and sufficient for primal/dual optimality (with zero duality gap).

In this case, we can use the KKT conditions to find a solution to our optimization problem, i.e., if we find (x^*, λ^*, ν^*) satisfying the conditions, we have found solutions to both the primal and the dual problems


$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$
$\mathbf{x}$
$(\boldsymbol{\lambda}, \boldsymbol{\nu})$

Soft-margin classifier:

$$
\begin{aligned}
min_{\mathrm{w},b,\xi} \quad & \frac{1}{2}\|\mathrm{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\
& \text{s.t. } y_i(\mathrm{w}^T x_i + b) \geq 1 - \xi_i, \qquad i = 1, ..., n \\
& \qquad\qquad \xi_i \geq 0 \qquad\quad i = 1, ..., n
\end{aligned}
$$

This optimization problem is differentiable and convex! That means :

- the KKT conditions and necessary and sufficient conditions for primal/dual optimality (with zero duality gap)
- we can use these conditions to find a relationship between the solutions of the primal and dual problems
- the dual optimization problem will be easy to “kernelize”

$$\begin{aligned} &min_{\mathrm{w},b,\xi} \qquad \frac{1}{2}\|\mathrm{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ &\text{s.t.} \qquad 1-\xi_i - y_i(\mathrm{w}^T x_i + b) \le 0, \qquad i = 1, ..., n \\ &\qquad\qquad\qquad\qquad -\xi_i \le 0 \qquad i = 1, ..., n \end{aligned}$$

The Lagrangian function is then given by

$$\begin{aligned} L(\mathrm{w}, b, \xi, \alpha, \beta) =& \frac{1}{2}\mathrm{w}^T\mathrm{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i + \\ & \sum_{i=1}^{n}\alpha_i \ (1-\xi_i - y_i(\mathrm{w}^T x_i + b)) - \sum_{i=1}^{n}\beta_i \ \xi_i \end{aligned}$$

Soft-margin dual

The Lagrangian dual is thus

$$L_D(\alpha, \beta) = min_{\mathrm{w},b,\xi} \ \ L(\mathrm{w}, b, \xi, \alpha, \beta)$$

and the dual optimization problem is

$$\max_{\alpha,\beta:\alpha_i,\beta_i \ge 0} \ \ L_D(\alpha, \beta)$$

Let's calculate a simplified expression for $L_D(\alpha, \beta)$ – the way we are gonna do it using KKT condition

- by taking the gradient with respect to (w, b, ξ) and setting this equal to zero,
- and plugging the result back into the formula

Taking the gradient w.r.t primal variables and setting those to zero:

$$\begin{aligned} L(\mathrm{w}, b, \xi, \alpha, \beta) =& \frac{1}{2}\mathrm{w}^T\mathrm{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i + \\ & \sum_{i=1}^{n}\alpha_i \ (1-\xi_i - y_i(\mathrm{w}^T x_i + b)) - \sum_{i=1}^{n}\beta_i \ \xi_i \end{aligned}$$

$$\begin{aligned} \nabla_{\mathrm{w}} L(\mathrm{w}, b, \xi, \alpha, \beta) &= \mathrm{w} - \sum_{i=1}^{n}\alpha_i y_i x_i = 0 \\ \frac{\partial}{\partial b} L(\mathrm{w}, b, \xi, \alpha, \beta) &= -\sum_{i=1}^{n}\alpha_i y_i = 0 \\ \frac{\partial}{\partial \xi_i} &= \frac{C}{n} - \alpha_i - \beta_i = 0 \end{aligned}$$

Plugging this in, the dual function is:

$$L_D(\alpha, \beta) = -\frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i$$

and the dual optimization problem can be written as:

$$\begin{aligned} \max_{\alpha,\beta} \ & -\frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \\ \text{s.t.} \ & \sum_i \alpha_i y_i = 0 \\ & \alpha_i + \beta_i = \frac{C}{n} \quad \alpha_i, \beta_i \geq 0 \end{aligned}$$

We can eliminate β to obtain

$$\begin{aligned} \max_{\alpha} \ & -\frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \\ \text{s.t.} \ & \sum_i \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq \frac{C}{n} \quad i = 1, ..., n \end{aligned}$$

Note: Input patterns are only involved via **inner products**. So we can replace $x_i^T x_j$ with $k(x_i^T x_j)$, as long as $k(\ )$ is calculating inner product in some space between these two points, we will be fine.

Recovering w^*:

Given α^* (the solution to the soft-margin dual), we can recover the optimal w^* and b by using KKT conditions.

- From KKT condition 1, we know that

  $$\mathrm{w}^* - \sum_{i=1}^{n} \alpha_i^* y_i x_i \ = \ 0$$

  hence the optimal normal vector w is just a linear combination of our input patterns.

  $$\mathrm{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$$

- b is a little less obvious – we'll return to this.

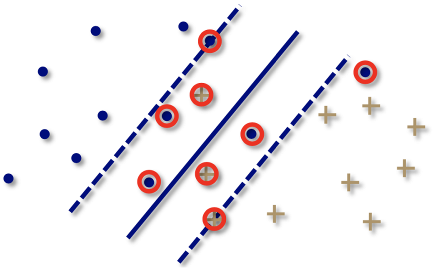
Support vectors

From KKT condition 5 (complementary slackness) we also have that for all i, either Lagrange multiplier or g_i has to be equal to zero:

$$\alpha_i^*(1 - \xi_i^* - y_i({w^*}^T x_i + b^*)) = 0$$

The x_i for which $y_i(w^* T x_i + b^*) = 1 - \xi_i^*$ are called **support vectors**.

x_i's that have non zero weights in the sum, are exactly the vectors that live inside the margin–these are the points on or inside the margin of separation.



By the KKT conditions, $\alpha_i^* \neq 0$ **iff** x_i is a support vector!

It has been empirically demonstrated that in typical learning problems, only a small fraction of the training input patterns are support vectors. Intuitively, if you doing a good job of separating two classes, not everything should be next to the boundary–so this empirical fact makes sense. if everything lives in the boundary, maybe you are not doing a good job of separating that data. So typically when things are working well, only a small fraction of your training data is going to factor into this. And this gives us some nice properties, one of them is that:

- Support vector machines produce a hyperplane with a **sparse** representation.

  $$\mathrm{w}^* = \sum_{\text{support vectors}} \alpha_i^* y_i x_i$$

- Finding b^*: Another consequence of the KKT conditions (condition 5) is that $\forall i$, $\beta_i^* \xi_i^* = 0$. Since

  $$\alpha_i^* + \beta_i^* = \frac{C}{n}$$

  this implies that if $\alpha_i^* < \frac{C}{n}$, then this means that β_i^* has to be nonzero. Then by complementary slackness, $\xi_i^* = 0$.

Then that means that if we can find a single Lagrange multiplier, single α_i in our solution that is greater than 0 but less than C/n, then for that i we can write this equation:

Recall that if $\alpha_i^* > 0$ we also have that x_i is a support vector, and hence,

$$y_i(\mathrm{w}^{*T} x_i + b^*) = 1 - \xi_i^*$$

For any i such that $0 < \alpha_i^* < \frac{C}{n}$, we have

$$\begin{aligned} y_i(\mathrm{w}^{*T} x_i + b^*) &= 1 \\ &\downarrow \\ b^* &= y_i - \mathrm{w}^{*T} x_i \end{aligned}$$

In practice, it is common to average over several such counter numerical imprecision.

Support Vector Machines

Given an inner product kernel k, we can write the SVM classifier as:

$$f(x) = sign\left(\sum_i \alpha_i^* y_i k(x, x_i) + b^*\right)$$

where α^* is the solution of

$$\begin{aligned} \max_\alpha \quad & -\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + \sum_i \alpha_i \\ \text{s.t.} \quad & \sum_i \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq \frac{C}{n} \quad i = 1, ..., n \end{aligned}$$

and $b^* = y_i - \sum_j \alpha_j^* y_j k(x_i, x_i)$ for some i such that $0 < \alpha_i^* < \frac{C}{n}$. Notice that C was the penalty term and if we make C very small, it is going to limit the influence of outliers.

Some Remarks

- The final classifier depends only on the training vectors x_i with $\alpha_i > 0$, i.e., the **support vectors**
- The size (number of variables) of dual QP is n (remember we started at d dimensional space)–we have one Lagrange multiplier for each x_i. This also means it is independent of input space d, and independent of the kernel k, the mapping ϕ, or the space $\mathcal{F}$
  - remarkable, since the dimension of $\mathcal{F}$ can be **infinite**.

  We don't even need the space we are mapping our data to be finite!
- The soft-margin hyperplane was the first machine learning algorithm to be "kernelized", but since then the idea has been applied to many, many other algorithms.

- kernel ridge regression
- kernel PCA
- ...

Solving the quadratic program

How can we actually compute the solution to

$$\begin{aligned} \max_\alpha \quad & -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j q_{ij} + \sum_i \alpha_i \\ \text{s.t.} \quad & \sum_i \alpha_i y_i = 0, \quad 0 \le \alpha_i \le \frac{C}{n} \quad i = 1, ..., n \end{aligned}$$

where $q_{ij} := y_i y_j k(x_i, x_j)$?

There are several approaches to solving quadratic programs, and many can be applied to solve the SVM-dual. We will focus on a particular example that is very efficient and capitalizes on some of the unique structure in the SVM dual, called sequential minimal optimization (SMO).

Sequential minimal optimization

SMO is an example of an iterative **decomposition** algorithm

Initialize: $\alpha = 0$

Repeat until stopping criteria satisfied:

1. Select a pair $i, j, 1 \le i, j \le n$
2. Update α_i and α_j by optimizing the dual QP, holding off all other $\alpha_k, k \neq i, j$ fixed

The reason for decomposing this into a two-variable subproblem is that this subproblem can be solved exactly via simple analytical update.

The update step: Choose α_i and α_j to solve

$$\begin{aligned} \max_{\alpha_i,\alpha_j} \quad & -\frac{1}{2}\left(\alpha_i^2 q_{ii} + \alpha_j^2 q_{jj} + 2\alpha_i\alpha_j q_{ij}\right) + c_i\alpha_i + c_j\alpha_j \\ \text{s.t.} \ & \alpha_i y_i + \alpha_j y_j = -\sum_{k\neq i,j}\alpha_k y_k \\ & 0 \le \alpha_i, \alpha_j \le \frac{C}{n} \end{aligned}$$

where $c_i = 1 - \frac{1}{2}\sum_{k\neq i,j}\alpha_k q_{ik}$ and similarly for c_j

In practice,

- Several strategies have been proposed for selecting (i, j) at each iteration
- typically based on heuristics (often using the KKT conditions) that predict which pair of variables will lead to the largest change in the objective function
- For many of these heuristics, the SMO algorithm is proven to converge to the global optimum after finitely many iterations
- The run-time complexity is $O(n^3)$ in the worst case and on average more like $O(n^2)$.

Alternative algorithms

SMO is one of the predominant strategies for training SVM, but there are important alternatives to consider on very large datasets.

- modern variants for solving the dual based on stochastic gradient descent–very closely related to SMO.
- directly optimizing the primal
  - makes most sense when the dimension of the feature space is small compared to the size of the dataset
  - some algorithms are very similar to PLA and stochastic gradient descent version of logistic regression

A general approach to "kernalization"

In order to "kernelize" an algorithm, the general approach consists of three main steps:

1. Show that the training process only involves the training data via inner products (i.e., $\mathrm{x}_i^T\ \mathrm{x}_j$)
2. Show that applying the decision rule to the new x only involves computing inner products (i.e., $\mathrm{w}^T\ \mathrm{x}$)
3. Replace all inner products with evaluations of the kernel function $k(\cdot, \cdot)$

This approach extends well beyond SVMs.