# General

- SVM is fundamentally a linear model in the way of defining margins and hyperplanes. However, it has great advantage when handling nonlinear problems by using kernel trick, which essentially maps the input features to a high-dimensional space. In generalized linear models (GLMs), similar feature mapping can be used to allow GLMs handle nonlinear problems, although not as efficient as SVMs.
- Support vectors are the data points that lie closest to the decision surface (or hyperplane). They are the data points most difficult to classify.
- SVMs maximize the margin around the separating hyperplane, which is **fully specified by a (usually very small) subset of training samples**, the support vectors.

# Large margin

In logistic regression, $h_\theta(x) = g(\theta^T x)$ is predicted as 1 when $\theta^T x \geq 0$. To predict $y = 1$ with more confidence or with large margin, we require $\theta^T x \gg 0$.

# Notation

Adopt a new notation $h_{w,b}(x) = g(wx + b)$. Also $y \in \{0, 1\}$ is changed to $y \in \{-1, 1\}$. Here, $g(z) = 1$ if $z \geq 0$, and $g(z) = -1$ otherwise. From this definition of $g$, the classifier will directly predict either 1 or −1 (cf. the perceptron algorithm), without first going through the intermediate step of estimating the probability of $y$ being 1 (which was what logistic regression did). In logistic regression, we need for example predict $y = 1$ when $g(z) \geq 0.5$.

# Review of normal vector, gradient, decision boundary

## Normal vector

First not confuse the following concepts such as normal vector, unit normal vector, vector norm, normalized vector.
From http://mathworld.wolfram.com/NormalVector.html (http://mathworld.wolfram.com/NormalVector.html), if a surface is written as an implicit function as $f(x, y, z) = 0$, then the normal vector at a $(x_0, y_0, z_0)$ is $\nabla_f(x_0, y_0, z_0)$. Imagine a sphere, then this normal corresponding to a direction pointing from the origin to the outside of the sphere.

If the surface above is written explicitly as $z = f(x, y)$, then the normal vector at a point $(x_0, y_0)$ on a surface is given by $[f_x(x_0, y_0), f_y(x_0, y_0), -1]$. This can easily be derived from the implicit case. If a surface is written as an implicit function $f(x, y, z) = ax + by + cz + d = 0$, then the normal vector is given as $[f_x(x_0, y_0), f_y(x_0, y_0), -1]$.

For planar surface, $f(x, y, z) = ax + by + cz + d = 0$, then we can obtain its normal $(a, b, c)$ from either implicit or explicit format. This normal vector is same when going through any point $(x_0, y_0)$ of the plane.

The plane with a normal vector $n = (a, b, c)$ and going through $r = (x_0, y_0, z_0)$ can be described by $n \cdot r = 0$. This is different from earlier discussed case where normal vector goes through the point $r = (x_0, y_0, z_0)$.

**Be careful about the explicit and implicit description of a function. An example below:**

- For $y = x^2$, the gradient we normally referred to here is just the 1D derivative $y = 2x$ and its direction is along $x$ axis but not the tangential direction.
- For $f(x, y) = x^2 - y$, the gradient is $(f_x, f_y) = (2x, -1)$. At $x = 1$, the direction of gradient is $(2, -1)$, which is pointing outward from the parabola.

## Decision boundary

Take logistic regression for example, we can try to solve for $p(x) = \frac{1}{2}$ and obtain a decision boundary $w^T x = 0$. This is a linear decision boundary. Note it is not that whenever we have a $w^T x$ in the hypothesis function and then we have a linear decision boundary. We must solve the specific equation and obtain the relation between $w$ and $x$. In fact, for many nonlinear classifiers, we cannot obtain a closed form solution. In that case, we need just plot the output of the classifier with a contour plot. See the decision boundary derivation for logistic regression in the following link https://stats.stackexchange.com/questions/136173/formula-for-decision-boundary-of-a-classifier-in-order-to-visualize-it (https://stats.stackexchange.com/questions/136173/formula-for-decision-boundary-of-a-classifier-in-order-to-visualize-it).

# Functional and geometric margins

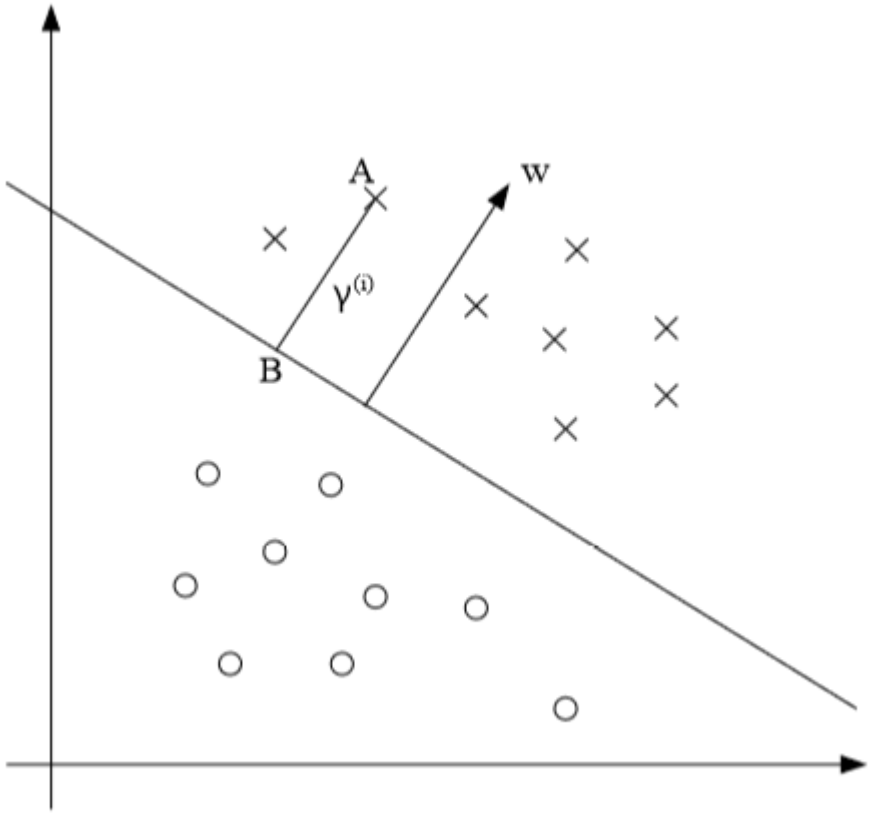
## Functional margin

Given a training example $(x^{(i)}, y^{(i)})$, we define the functional margin of $(w, b)$ with respect to the training example:

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$$

If $y^{(i)} = 1$, then for the functional margin to be large (i.e., for our prediction to be confident and correct), we need $(w^T x + b)$ to be a large positive number. Conversely, if $y^{(i)} = -1$, then for the functional margin to be large, we need $(w^T x + b)$ to be a large negative number. Moreover, if $y^{(i)}(w^T x + b) > 0$, then our prediction on this example is correct. Hence, a large functional margin represents a confident and a correct prediction.

For a linear classifier with the choice of $g$ given above (taking values in {−1,1}), there is one property of the functional margin that makes it not a very good measure of confidence. Given our choice of g, we note that if we replace $w$ with $2w$ and $b$ with $2b$, then since $g(w^T x + b) = g(2w^T x + 2b)$, this would not change $h_{w,b}(x)$ at all. $g$, and hence also $h_{w,b}(x)$, depend only on the sign, but not on the magnitude, of $w^T x + b$. Note the $g$ here will not give different probability values as $w^T x + b$ changes whenever it keeps the same sign. This is not like the case of logistic regression. However, replacing $(w, b)$ with $(2w, 2b)$ also results in multiplying our functional margin by a factor of 2. Thus, it seems that by exploiting our freedom to scale w and b, we can make the functional margin arbitrarily large without really changing anything meaningful. Intuitively, it might therefore make sense to impose some sort of normalization condition.



## Geometric margins

In the figure above, the distance between sample A to the decision boundary can be solved as (see details on cs229 notes)

$$\gamma^{(i)} = \frac{w^T x^{(i)} + b}{\|w\|} = \left( \frac{w}{\|w\|} \right)^T + \frac{b}{\|w\|}$$

Now define a geometric margin as

$$\gamma^{(i)} = y^{(i)} \left( \left( \frac{w}{\|w\|} \right)^T + \frac{b}{\|w\|} \right),$$

where the introduction of $y^{(i)}$ makes the relation hold in both sides of the decision boundary. If $\|w\| = 1$, then geometric margin is same as the functional margin. Connecting this definition and the figure above, we understand why this is called a geometric margin. The bigger the 'vertical' distance between sample A and the decision boundary, the bigger the margin.

The geometric margin is invariant to rescaling of the parameters; i.e., if we replace $w$ with $2w$ and $b$ with $2b$, then the geometric margin does not change. This will in fact come in handy later. Specifically, because of this invariance to the scaling of the parameters, when trying to fit $w$ and $b$ to training data, we can impose an arbitrary scaling constraint on $w$ without changing anything important; for instance, we can demand that $\|w\| = 1$, or $\|w_1\| = 5$, or $\|w_1 + b\| + \|w_2\| = 2$ , and any of these can be satisfied simply by rescaling $w$ and $b$.

Finally, define the following two quantities for a training set

$$\hat{\gamma} = \min_{i=1,2,...m} \hat{\gamma}^{(i)}$$
$$\gamma = \min_{i=1,2,...m} \gamma^{(i)}$$

# The optimal margin classifier

Recall that PCA maximize the variance by choosing a specific direction (axis). Here SVM maximizes the margins. The first and most natural optimization problem is to (**maximize the minimum margin?**)

$$\max_{\gamma,w,b} \quad \gamma$$
$$\text{s. t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq \gamma, i = 1, 2, \ldots m$$
$$\|w\| = 1$$

The $\|w\| = 1$ constraint ensures that the functional margin equals to the geometric margin, so we are also guaranteed that all the geometric margins are at least $\gamma$. However, this problem cannot be solved easily due to the non-convex nature of $\|w\| = 1$. Therefore, transform the problem to be a nicer one

$$\max_{\hat{\gamma},w,b} \quad \frac{\hat{\gamma}}{\|w\|}$$
$$\text{s. t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma}, i = 1, 2, \ldots m$$

Although the $\|w\| = 1$ is gone, the non-convext $\|w\|$ appears in $\frac{\hat{\gamma}}{\|w\|}$, and it is thus still hard to solve. Recall the earlier discussion that an arbitrary scaling constraint on $w, b$ has no effect on the final result. We thus introduce the scaling constraint which can make $\hat{\gamma} = 1$. The previous problem then becomes

$$\begin{aligned} \max_{\gamma,w,b} \quad & \frac{1}{\|w\|} \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \dots m \end{aligned}$$

Because maximizing $\frac{1}{\|w\|}$ is same as minimize $\|w\|^2$, finally obtain the following optimization problem

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \dots m \end{aligned}$$

**Should the symbol $\gamma$ gone in the above equation?**

# Solving the optimization problem

The above optimization problem (minimizing quadratic equation) can be done directly by using commercial quadratic programming (QP) code. This involves linear programming in numerical linear algebra. However, here we will resort to another way and solve the dual problem of the original(primal) problem. The reason is that solving the dual problem can allow us employ the kernel trick, and thus calculate efficiently when handling very high-dimensional feature space.

## Lagrange duality

General idea: The primal form is just the original optimization problem where we max first and then min. The dual form is, however, reverse the order. Under some conditions, the two forms can give the same solution.
Unlike Lagrange multiplier method, usually used for the normal optimization problem with equality constraints, we need generalize this to inequality constraints. About the intuition of Lagrange multiplier, check the parallel gradient condition in the MIT notes for SVM.
Consider the primal optimization problem

$$\begin{aligned} \min_{w} \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0, i = 1, 2, \dots k \\ & h_i(w) = 0, i = 1, 2, \dots l. \end{aligned}$$

To solve this problem, introduce the following the generalized Lagrangian

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

where $\alpha_i, \beta_i$ are Lagrange multipliers. Consider the quantity $\theta_p(w) = \max_{\alpha,\beta:\alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$. It can be shown that $\theta_p(w) = f(w)$ when $w$ satisfy the primal constraints. Hence, if we consider the minimization problem

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

we see that it is the same problem (i.e., and has the same solutions as) our original, primal problem.
Now, consider a slightly different problem. We define $\theta_D(\alpha, \beta) = \min_w \mathcal{L}(w, \alpha, \beta)$. We can now pose the dual optimization problem:

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

This is exactly the same as our primal problem shown above, except that the order of the 'max' and the 'min' are now exchanged. Normally we have

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

However, under certain conditions, $p^* = d^*$.
Suppose $f$ and the $g_i$ are convex and the $h_i$ are affine. Suppose further that the constraints $g_i$ are (strictly) feasible; this means that there exists some $w$ so that $g_i(w) < 0$ for all $i$. Under these assumptions, there must exist $w^*, \alpha^*, \beta^*$ so that $w^*$ is the solution to the primal problem,$\alpha^*, \beta^*$ are the solution to the dual problem. Moreover, $w^*, \alpha^*, \beta^*$ satisfy the Karush-Kuhn-Tucker (KKT) conditions, which are as follows:

$$\frac{\partial}{\partial w_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, i = 1, 2, \ldots n$$

$$\frac{\partial}{\partial \beta_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, i = 1, 2, \ldots l$$

$$\alpha_i^* g_i(w^*) = 0, i = 1, 2, \ldots k$$

$$g_i(w^*) \leq 0, i = 1, 2, \ldots k$$

$$\alpha^* \geq 0, i = 1, 2, \ldots k$$

Conversely, if $(w^*, \alpha^*, \beta^*)$ satisfy the above KKT conditions, then they are solutions to both primal and dual problems.
We draw attention to the equation $\alpha_i^* g_i(w^*) = 0$, which is called the KKT dual complementarity condition. It implies that if $\alpha_i^* > 0$, then $g_i(w^*) = 0$. That is, the $g_i(w^*) \leq 0$ constraint is active, meaning it holds with equality rather than with inequality. This is the key for showing that the SVM has only a small number of support vectors. Moreover,the KKT dual complementarity condition will also give us our convergence test when we talk about the SMO algorithm.
Some notes on convex and affine :

- When $f$ has a Hessian, then it is convex if and only if the Hessian is positive semidefinite. For instance, $f(w) = w^T w$ is convex; similarly, all linear (and affine ) functions are also convex. A function $f$ can also be convex without being differentiable, but we won't need those more general definitions of convexity here.
- $h_i$ is affine means there exists $a_i, b_i$, so that $h_i(w) = a_i^T w + b_i$. 'Affine' means the same thing as linear, except that we also allow the extra intercept term $b_i$.

# Lagrange duality applied to SVM

The following is the optimization problem for SVM

$$\min_{\gamma,w,b} \quad \frac{1}{2}\|w\|^2$$

$$\text{s. t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \ldots m$$

We can write the constraints as $g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leq 0$. Note that from the KKT dual complementarity condition, we will have $\alpha_i > 0$ only for the training examples that have functional margin exactly equal to one (i.e., the ones. corresponding to constraints that hold with equality, $g_i(w) = 0$). The following are two slides for an intuitive understanding. From the slides, we know that there are usually only a small number of samples satisfying for the constraints $g_i(w) = 0$. Therefore, only a small number of $\alpha_i$ (the Lagrange multipliers) will be bigger than zero. The $g_i(w) = 0$ condition are just the $w^T x + b = 1$ and $w^T x + b = -1$ conditions.

# Definitions

Define the hyperplanes $H$ such that:

$w \bullet x_i + b \geq +1$ when $y_i = +1$

$w \bullet x_i + b \leq -1$ when $y_i = -1$

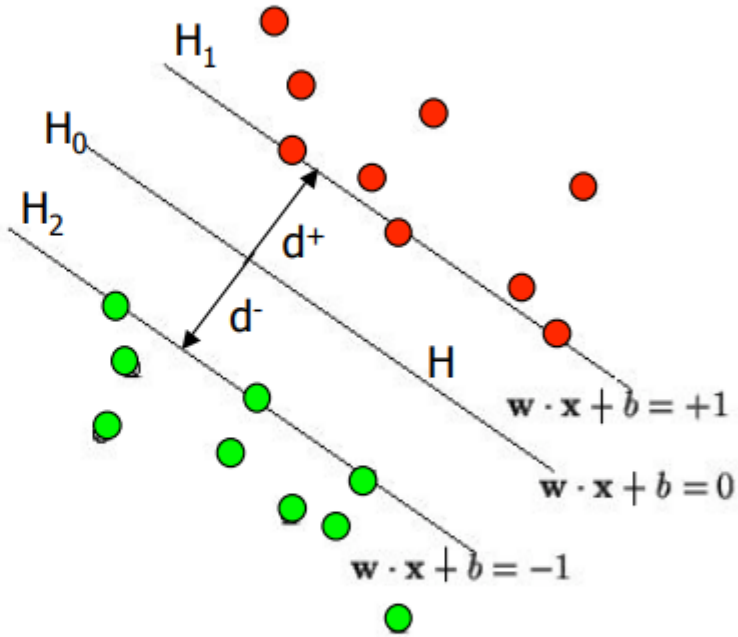
$H_1$ and $H_2$ are the planes:

$H_1$: $w \bullet x_i + b = +1$

$H_2$: $w \bullet x_i + b = -1$

The points on the planes $H_1$ and $H_2$ are the tips of the Support Vectors

The plane $H_0$ is the median in between, where $w \bullet x_i + b = 0$




d+ = the shortest distance to the closest positive point

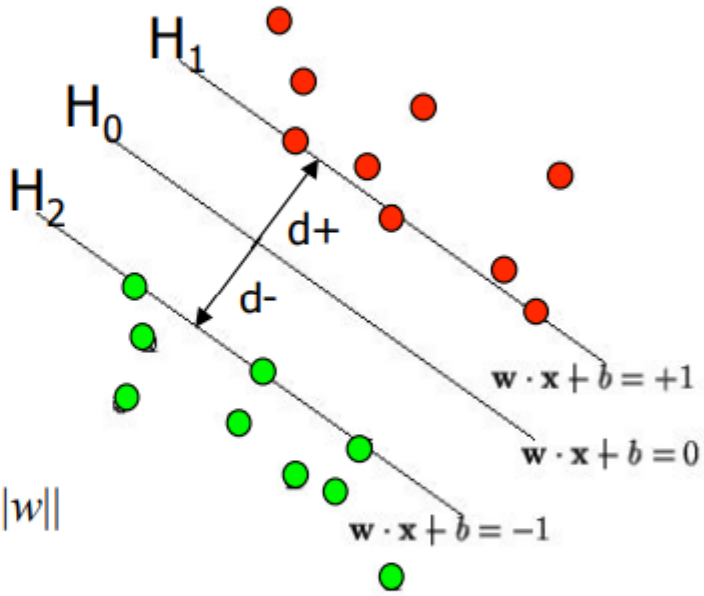
d- = the shortest distance to the closest negative point

The margin (gutter) of a separating hyperplane is d+ + d–.

Check how the margins are defined in the slide. We are supposed to maximize only the margins related to support vectors but not the margins from all other samples.

# Maximizing the margin (aka street width)

We want a classifier (linear separator) with as big a margin as possible.




Recall the distance from a point$(x_0,y_0)$ to a line: $Ax+By+c = 0$ is: $|Ax_0 +By_0 +c|/\text{sqrt}(A^2+B^2)$, so,
The distance between $H_0$ and $H_1$ is then: $|w\bullet x+b|/||w||=1/||w||$, so

The total distance between $H_1$ and $H_2$ is thus: $2/||w||$

In order to maximize the margin, we thus need to minimize $||w||$. With the condition that there are no datapoints between $H_1$ and $H_2$:

$\mathbf{x}_i\bullet\mathbf{w}+b \geq +1$ when $y_i = +1$
$\mathbf{x}_i\bullet\mathbf{w}+b \leq -1$ when $y_i = -1$

**Can be combined into: $y_i(\mathbf{x}_i\bullet\mathbf{w}) \geq 1$**

Now we begin to solve the problem. First construct the Lagrangian for the optimization problem

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

where there are only $\alpha_i$ but no $\beta_i$ Lagrange multipliers, since the problem has only inequality constraints. In order to introduce the kernel trick, we will solve the dual form of the original primal problem. So we need first find the dual form of the problem. To do so, we need to first minimize $\mathcal{L}(w, b, \alpha)$ with respect to $w$ and $b$ (for fixed $\alpha$), to get $\theta_D$. We do this by setting the derivatives of $\mathcal{L}$ with respect to $w$ and $b$ to zero and obtain respectively As for the derivative with respect to b, we obtain

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

Plugging these into the previous Lagrangian gives

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

where $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$ has been used in the derivation. With the constraints, we finally have

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ \text{s. t.} \quad & 0 \leq \alpha_i \geq 0, i = 1, 2, \ldots m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned}$$

We can show that the conditions required for $p^* = d^*$and the KKT conditions are indeed satisfied in the optimization problem. Hence, we can solve the dual in lieu of solving the primal problem. Once we calculated $\alpha_i$, then we can calculate $w$ and $b$. And thus using the following equation to calculate

$$w^T x + b = \left( \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T x + b$$

, if its value is bigger than zero, then we predict $y = 1$. Because they are usually a small number of $\alpha_i > 0$, many terms in the sum are zero.

# Regularization and the non-separable case

The original optimization problem is modified as

$$\min_{\gamma,w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$$
$$\text{s. t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, 2, \ldots m$$
$$\xi_i \geq 0, i = 1, 2, \ldots m$$

Here $l_1$ regularization is used. Examples are now permitted to have functional margin less than 1, and if an example has a functional margin $1 - \xi_i$ ($\xi > 0$),we would pay a cost of the objective function being increased by $C\xi_i$. The parameter C controls the relative weighting between the twin goals of making the $\|w\|^2$ small (which we saw earlier makes the margin large) and of ensuring that most examples have functional margin at least 1. For this problem, we can form a new Lagrangian and derive its dual form, and finally obtain the following optimization problem

$$\max_{\alpha} \quad W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)}\rangle$$
$$\text{s. t.} \quad 0 \leq \alpha_i \leq C, i = 1, 2, \ldots m$$
$$\sum_{i=1}^{m}\alpha_i y^{(i)} = 0$$

With $l_1$ regularization, one change is from $\alpha_i \geq 0$ to $0 \leq \alpha_i \leq C$. The other change is the calculation of $b$ (see details in Platt's paper). Finally, the KKT complementarity conditions are:

$$\alpha_i = 0 \Rightarrow y^{(i)}(w^T x^{(i)} + b) \geq 1$$
$$\alpha_i = C \Rightarrow y^{(i)}(w^T x^{(i)} + b) \leq 1$$
$$0 < \alpha_i < C \Rightarrow y^{(i)}(w^T x^{(i)} + b) = 1$$

**The above equations are obtained with $l_1$ regularization. In the problem set, there is $l_2$ regularization.**

# Sequential minimal optimization (SMO) algorithm

The SMO algorithm gives an efficient way of solving the dual problem arising from the derivation of the SVM. Partly to motivate the SMO algorithm, and partly because its interesting in its own right, let's first take another digression to talk about the coordinate ascent algorithm.

## Coordinate ascent algorithm

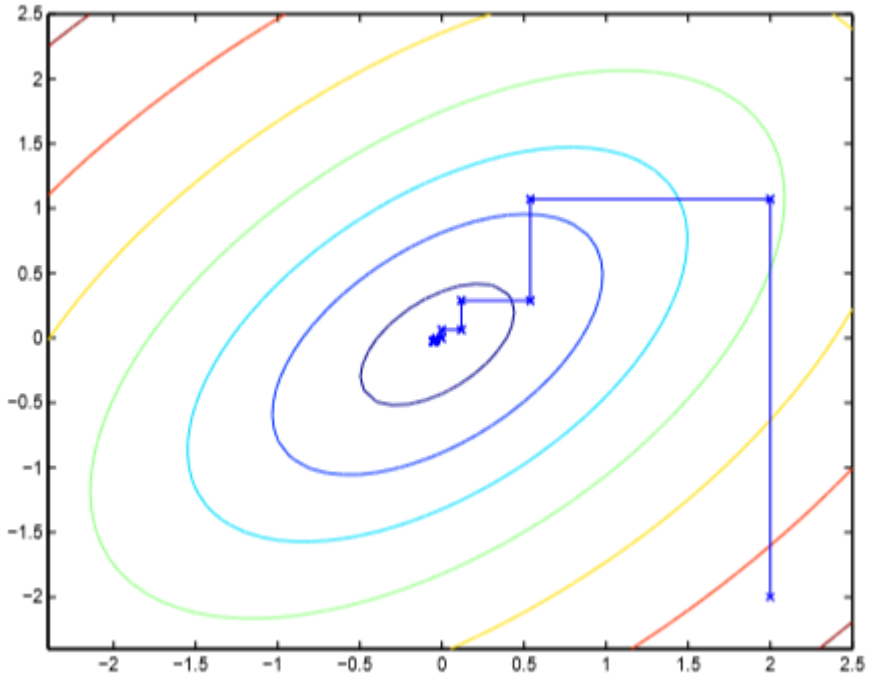
Consider the following problem

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

For the above optimization problem, two conventional methods are respectively **gradient ascent and Newton's method**. The coordinate ascent method is the third one, which loops over from 1 to $m$ the following equation until converged.

$$\alpha_i := \max_{\hat{\alpha_i}} W(\alpha_1, \alpha_2, \ldots \alpha_{i-1}, \hat{\alpha_i}, \alpha_{i+1} \alpha_m)$$

The idea is optimize along one axis (one $\alpha$) while keeping all other variables fixed. A two-variable cartoon is shown below to show the process.




When the function $W$ happens to be of such a form that the "argmax" in the inner loop can be performed efficiently, then coordinate ascent can be a fairly efficient algorithm. Otherwise coordinate ascent is less efficient than newton method.

## SMO derivation

Now consider the optimization problem derived with $l_1$ regularization earlier. Due to the constraints in the problem, not all the $\alpha_i$ are independent. So we cannot update $\alpha$ one by one like in the typical coordinate ascent algorithm, but need update at least two simultaneously. Or may be understood we still update one by one for not all the $\alpha$, but only the independent $m - 1$ of them. Here is the process:

Repeat till convergence

- Select some pair $\alpha_i$ and $\alpha_j$ 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).
- Reoptimize $W(\alpha)$ with respect to $\alpha_i$ and $\alpha_j$, while holding all the other $\alpha_k, k \neq i, j$ fixed.

To test for convergence of this algorithm, we can check whether the previously given KKT complementary conditions are satisfied to within some tol. Here, tol is the convergence tolerance parameter, and is typically set to around 0.01 to 0.001. (See Platt paper and pseudocode for details.)

In each step, if optimal point is within the box, then it is done. If not, then clip to the boundary. The key element making us to use coordinate ascent efficiently is in each step, the update can be calculated with a closed form solution. Here, it means $\alpha$ can be calculated directly from a simple quadratic equation.

## Other ways for optimizing SVM equations

- See other ways in SVM entry of Wikipedia.
- See how the landmarks are assigned in notes of Coursera.

Note in the dual problem without linearly separable hypothesis, we only have a small change from 0<=alpha to 0 <=alpha<=C. All other kesi, ri, etc are gone. How?

Understand intuitively how alpha will give separated if maximized. Some derivation of SMO will be in the homework.

# Comparison to other algorithm

# Kernels

The original maximum-margin hyperplane algorithm proposed by Vapnik in 1963 constructed a linear classifier. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick (originally proposed by Aizerman et al.) to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional; although the **classifier is a hyperplane in the transformed feature space, it may be nonlinear in the original input space.**

# Polynomial kernel

Assume $x, z \in \mathbb{R}^n$, and we want to extend the feature space from e.g. $(x_1, x_2, x_3)$ to $\phi(x) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2 \ldots, x_3x_3)$for the case of $n = 3$. This is a tempt to include second-order nonlinear terms to construct a non-linear classification. This is same as the trick of extending linear regression to polynomial regression.
In SVM, we need calculate inner product $x^Tz$. After the above feature mapping, to calculate the inner product $\phi^T(x)\phi(z)$, we need a time complexity of $O(n^2)$. In the above example, we need calculate $n^2 = 3^2 = 9$ terms in evaluating $\phi(x)$.
Now introducing a kernel trick. We will not calculate $\phi(x)$ explicitly to calculate the inner product, but calculate $K(x, z) = (x^Tz)^2$ (which have a time complexity of $O(n)$. It can be show that the easier calculated $K(x, z)$ (called kernel) is same as the $\phi^T(x)\phi(z)$. Thus a complexity $O(n^2)$ problem becomes a $O(n)$ problem.
In general, we can calculate $K(x, z) = (x^Tz + c)^d$, which corresponds to a mapping to a ${}_{n+d}C_d$ feature space, including all the monomials up to order $d$. Here the time complexity goes from $O(n^d)$ to $O(n)$.

**Comments:**

- Although in the beginning of original notes of cs229, $x$ is a scalar for housing area, it can be a vector.
- $\phi(x)$ is not a function although it looks like a function. It is a mapping but not the mapping of a function. For example $\phi(x)$ maps a single scalar to multiple values.
- Unlike the polynomial kernel where we can write out the explicit form of $\phi(x)$, most cases we cannot write the explicit form of $\phi(x)$, or it is very expensive to calculate $\phi(x)$. The key is we can efficiently calculate $\phi(x)^T\phi(x)$ without calculating $\phi(x)$, or even without knowing the form of $\phi(x)$.
- In the case of $K(x, z) = (x^Tz + c)^d$, the dimension of corresponding feature space of $\phi(x)$ is not $d$ dimension. The dimension also depends on the dimension of $x$.

# Gaussian kernel

Kernel is an inner product and thus measure similarity. From this intuition, Gaussian kernel is introduced as

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

When $x, z$ are similar, the kernel approaches 1. When they are far apart, then the kernel takes value 0.

# Mercer Theorem

Let $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be given. Then for $K$ to be a valid (Mercer) kernel, it is necessary and sufficient that for any $x^{(1)}, \ldots, x^{(m)}, m < \infty$, the corresponding kernel matrix is symmetric positive semi-definite.
Check p17 of cs229 notes about how to show kernel is positive semi definite.

# Choosing Kernels

- Use prior knowledge about whether the problem is a linear or nonlinear problem. For linear problem or with too many features, we may use linear kernel, or no kernel. For nonlinear problem, Gaussian kernel (redial basis function: RBF) might be a good default.

## Choosing Gaussian kernel

- If n is small and/or m is large e.g. 2D training set that's large. Gaussian kernel introduce infinite number of features. https://en.wikipedia.org/wiki/Radial_basis_function_kernel (https://en.wikipedia.org/wiki/Radial_basis_function_kernel). Large training set prevent overfit. However, if m is too large, then Gaussian kernel is slow in calculation.
- make sure you perform feature scaling before using a Gaussian kernel. Otherwise features with a large value will dominate the f value.
- Gaussian kernel is probably the most popular kernel (for nonlinear classification).
- For Gaussian kernel parameter ($\gamma$ or $\sigma$), if $\sigma$ is small, then only very close data points are similar. This has a higher resolution, and thus easier to overfit?

## Choosing no kernel or linear kernel

Predict $y = 1$ if $\theta^T x >= 0$. Get a standard linear classifier. If n is large and m is small then Lots of features, few examples not enough data - risk over-fitting in a high dimensional feature-space. **The Gaussian kernel and linear kernel above are the most popular kernels.**

## Choosing polynomial Kernel

- Not used that often.
- Usually performs worse than the Gaussian kernel.

## Choosing string kernel

- Used if input is text strings.
- Use for text classification.

### Choosing Chi-squared kernel.

### Choosing histogram intersection kernel.

## Comparison to neural nets

- Similarity: For a set of (input, output) pairs, the output is a set of weights w (or wi), one for each feature, whose linear combination predicts the value of y. The weights is not probability.
- Difference 1: SVM uses the optimization of maximizing the margin to reduce the number of weights that are nonzero to just a few that correspond to the important features that 'matters in deciding the separating line(hyperplane). These nonzero weights correspond to the support vectors (because they 'support' the separating hyperplane).
- Difference 2: SVM optimizes a paraboloid under constraints and hence does not have the problem of local minimum.

## Compare to logistic regression

- If n (features) is large vs. m (training set) e.g. text classification problem Feature vector dimension is 10 000 Training set is 10 - 1000. Then use logistic regression or SVM with a linear kernel.
  **Understanding:** The key advantage of kernel trick is to introduce high dimensional features without causing calculation burden. If there are already many features, then SVM might not be necessary.
- If n is small and m is intermediate n = 1 - 1000 m = 10 - 10 000. Gaussian kernel is good.
  **Understanding:** Gaussian kernel introduce high-dimensional features (infinity) and thus can overcome the shortage of small number of features. Also the number of samples are intermediate, it is unlikely to overfit.
- If n is small and m is large n = 1 - 1000 m = 50 000+. SVM will be slow to run with Gaussian kernel. In that case (1)manually create or add more features. (2) Use logistic regression or SVM with a linear kernel.
  **Understanding:** Although Gaussian kernel is good in expanding feature space, it is not efficient when the sample number is very big. Check how this happens in equations.
- Logistic regression and SVM with a linear kernel are pretty similar. (1) Do similar things.(2) Get similar performance.
- A lot of SVM's power is using diferent kernels to learn complex non-linear functions.
- For all these regimes a well designed NN should work. But, for some of these problems a NN might be slower - SVM well implemented would be faster.
  **Understanding:** The key reason is that in SVM we only calculate the weights for a small number of samples (support vectors), while neural net has to calculate the weights of all samples.
- SVM has a convex optimization problem - so you get a global minimum.

- The algorithm SVM is widely perceived a very powerful learning algorithm.
  **Understanding:** The key advantage of SVM, in my opinion, is using kernel trick to handle nonlinear classification. Although logistic regression, or other algorithms in the class of Generalized linear models (GLM) can also handle nonlinear regression/classification by introducing new features (e.g. from $x \rightarrow (x, x^2, x^3 \dots)$. However, this will be very expensive as more and more features are introduce. SVM, however, can introduce high dimension features without introducing big calculation burden. For $d$ dimensional polynomial kernel, the time complexity can go from $O(n^d)$ to $O(n)$. For nonlinear problems with not so big number of samples, SVM often has an advantage.