Support Vector Machines

Bingyu Wang, Virgil Pavlu

March 30, 2015
based on notes by Andrew Ng.

1 What's SVM

The original SVM algorithm was invented by Vladimir N. Vapnik[1] and the current standard incarnation (soft margin) was proposed by Corinna Cortes[2] and Vapnik in 1993 and published in 1995.

A support vector machine(SVM) constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.[3] In this notes, we will explain the intuition and then get the primal problem, and how to translate the primal problem to dual problem. We will apply kernel trick and SMO algorithms to solve the dual problem and get the hyperplane we want to separate the dataset. Give general idea about SVM and introduce the goal of this notes, what kind of problems and knowledge will be covered by this node.

In this note, one single SVM model is for two labels classification, whose label is $y \in \{-1, 1\}$. And the hyperplane we want to find to separate the two classes dataset is h, for which classifier, we use parameters w, b and we write our classifier as

$$h_{w,b}(x) = g(w^T x + b)$$

Here, $g(z) = 1$ if $z \geq 0$, and $g(z) = -1$ otherwise.

2 Margins

Following Andrew Ng[4], we will start the by talking about margins, which can give us the "confidence" of our predictions.

Consider logistic regression, where the probability $p(y = 1|x; w)$ is modeled by $h_w(x) = g(w^T x)$.We would then predict "1" on an input x if and only if $h_w(x) \geq 0.5$, or equivalently, if and only if $w^T x \geq 0$. Consider a positive training example ($y = 1$). The larger $w^T x$ is, the larger also is $h_w(x) = p(y = 1|x; w, b)$, and thus also the higher our degree of "confidence" that the label is 1. Thus informally we can think of our prediction as being a very confident one that $y = 1$ if $w^T x \gg 0$. Similarly, we think of logistic regression as making a very confident prediction of $y = 0$, if $w^T x \ll 0$. Given a training set, again informally it seems that we'd have found a good fit to the training data if we can find w so that $w^T x_i \gg 0$ whenever $y_i = 1$, and $w^T x_i \ll 0$ whenever $y_i = 0$, since this would reflect a very confident (and correct) set of classifications for all the training examples. This seems to be a nice goal to aim for, and we?ll soon formalize this idea using the notion of functional margins.

For a different type of intuition, consider the following Figure 1, in which x's represent positive training examples, o's denote negative training examples, a decision boundary (this is the line given by the equation $w^T x = 0$, and is also called the **separating hyperplane**) is also shown, and three points have also been labeled A, B and C.

[1] http://en.wikipedia.org/wiki/Vladimir_Vapnik

[2] http://en.wikipedia.org/wiki/Corinna_Cortes

[3] http://en.wikipedia.org/wiki/Support_vector_machine

[4] CS229 Lecture notes, Part V Support Vector Machines

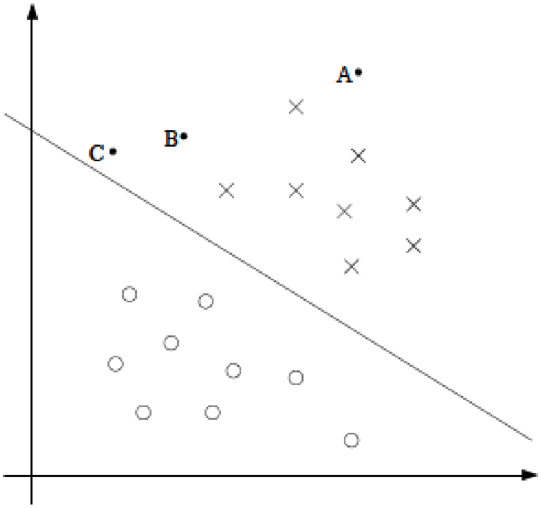

Figure 1: Confident Example, linearly separable.

Notice that the point A is very far from the decision boundary. If we are asked to make a prediction for the value of y at A, it seems we should be quite confident that $y = 1$ there. Conversely, the point C is very close to the decision boundary, and while it's on the side of the decision boundary on which we would predict $y = 1$, it seems likely that just a small change to the decision boundary could easily have caused our prediction to be $y = 0$. Hence, we're much more confident about our prediction at A than at C. The point B lies in-between these two cases, and more broadly, we see that if a point is far from the separating hyperplane, then we may be significantly more confident in our predictions. Again, informally we think it'd be nice if, given a training set, we manage to find a decision boundary that allows us to make all correct and confident (meaning far from the decision boundary) predictions on the training examples.

In another word, if we could find a decision boundary, who can give us a larger margin, it will be better than the one give us a smaller margin. From the following Figure 2, we can tell that the black decision boundary is better than the green decision boundary, because the black one gives us a larger margin than the green one.

2.1 Functional and Geometric Margins

Lets now formalize the margin intuition into notions of the functional and geometric margins. Given a training example (x_i, y_i), we define the functional margin of (w, b) with respect to the training example

$$\hat{\gamma}_i = y_i(w^T x + b)$$

Note that 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 (x_i, y_i) is correct. Hence, a large functional margin represents a confident and a correct prediction.

Given a training set $S = \{(x\_i, y\_i); i = 1, 2, \ldots, m\}$, we also define the function margin of (w, b) with respect to S to be the smallest of the functional margins of the individual training examples. Denoted by $\hat{\gamma}$, this can therefore be written:

$$\hat{\gamma} = \min_{i=1,\ldots,m} \hat{\gamma}_{(i)}$$

Functional margins can represent a confident and a correct prediction. The larger functional margins, the classifier better. However, by scaling w, b, we can make the functional margin arbitrarily large without really

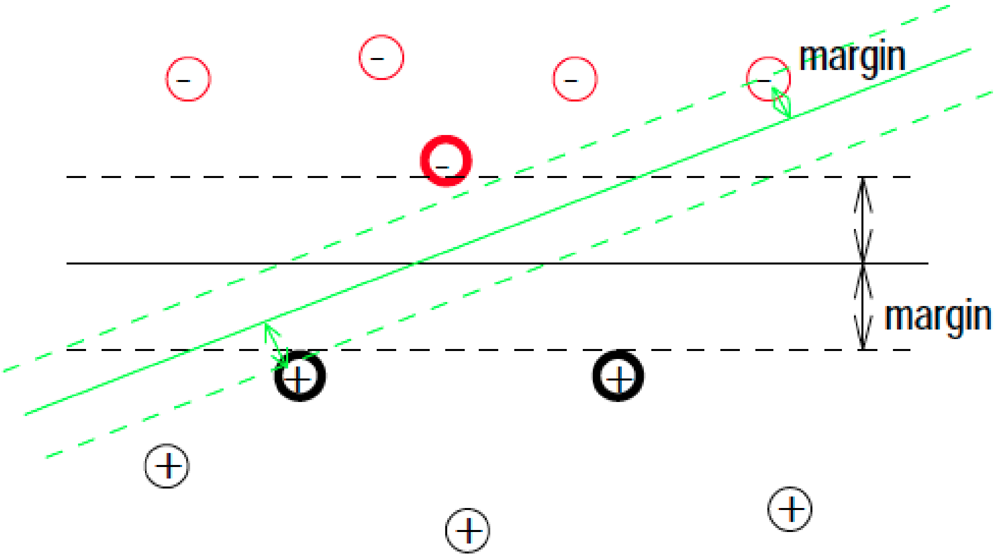

Figure 2: Margin Example. The black separating plane is better than the green one, because it has larger margins (sits more "in the middle"). A mechanical analogy: if the separating plane is free to rotate but constrained to be separator, when the points start pushing force towards the plane, the plane will settle in an equilibrium "middle" position - thats where the black separator is.

changing anything meaningful. Typically for a linear classifier, the final prediction is made by applying the sign function g to the linear score:

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

We note for any scalar c we can replace w with cw and b with cb, and have that $g(w^Tx+b) = g(cw^Tx+cb)$, this would not change the prediction $h_{w,b}(x) = g$ at all. I.e., g, and hence also $h_{w,b}(x)$, depends only one the sign, but not on the magnitude of $w^T + b$. However, replacing (w, b) with (cw, cb) also results in multiplying our functional margin by a factor of c. 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. We can make a reference decision on scale, and will choose the scale such that minimum functional margin is $y(w^Tx+b) = 1$.

2.1.1 Geometric Margins

In Figure 3, the decision boundary corresponding to (w, b) is shown, along with the vector w. Note that w is orthogonal to the separating hyperplane.[5]. Consider the opposing points at x_1 and x_2 which represents training examples closest to each other with labels $y_1 = 1$, $y_2 = -1$. The distance to the decision boundary, or the geometric margin ρ, is half of line segment x_1x_2, one minimum margin on each side.

As the picture shows, we have scaled (by a constant) w and b such that the closest points are on the line $|w^Tx + b| = 1$. This is a reference decision, and it can be done since any scalar c applied to w, b does not change the plane : $cw^Tx + cb = 0$ is the same plane as $w^Tx + b = 0$. We can write this constraint, that all points are no closer than lines $|w^Tx + b| = 1$ (either side of the plane) by using the labels for signs: $y(w^Tx + b) \geq 1$. In other words, the constraints state that all functional margins are at least 1.

From $w^Tx_1 + b = 1$ and $w^Tx_2 + b = -1$ we have that $w^T(x_1 - x_2) = 2$; considering that w and $x_1 - x_2$ are parallel vectors, we obtain that $||x_1 - x_2|| = 2/||w||$. Since $||x_1 - x_2|| = 2\rho$, we obtain that the minimum geometric margin is $\rho = 1/||w||$.

[5] http://mathworld.wolfram.com/NormalVector.html

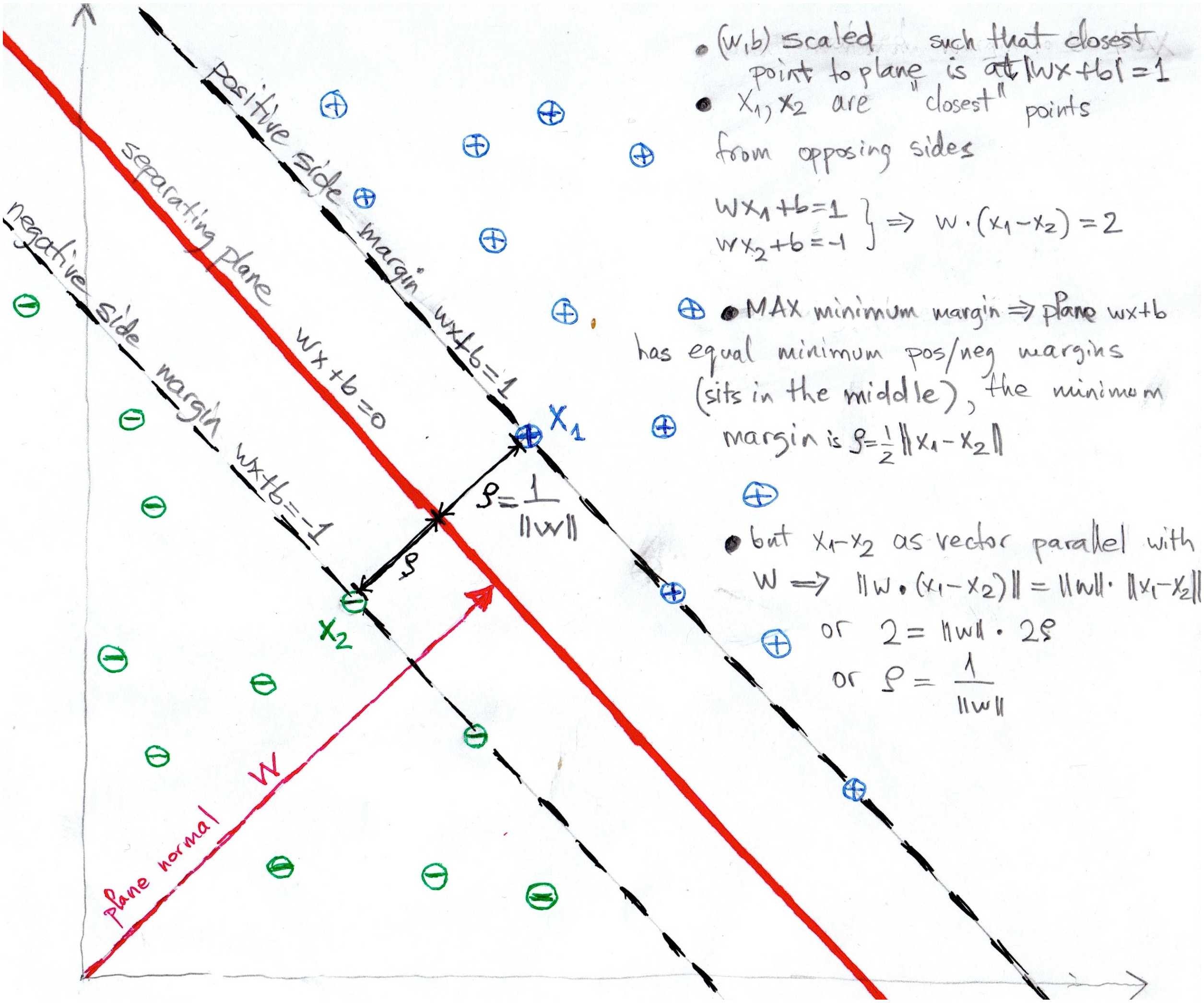
• (w,b) scaled such that closest point to plane is at |wx+b| = 1
• x1, x2 are "closest" points from opposing sides
wx1+b=1
wx2+b=-1
⇒ w·(x1−x2)=2
• MAX minimum margin ⇒ plane wx+b has equal minimum pos/neg margins (sits in the middle), the minimum margin is ρ = 1/2 ||x1 − x2||
• but x1−x2 as vector parallel with w ⇒ ||w·(x1−x2)|| = ||w||·||x1−x2||
or 2 = ||w||·2ρ
or ρ = 1/||w||
positive side margin
wx+b=1
separating plane
wx+b=0
negative side margin
wx+b=-1
X1
X2
ρ = 1/||w||
ρ
W
plane normal

Figure 3: Geometric Margin: w and b are scaled such that closest points are on the line $|w^T x + b| = 1$. If the plane is in the middle, the minimum margin (geometrical distance from plane to points) is $\rho = 1/w$.

3 The Optimal Margin Classifier

From the intuition of margins before, we try to find a decision boundary that maximizes the geometric margin, since this would reflect a very confident set of predictions on the training set and a good "fit" to the training data. Specifically, this will result in a classifier that separates the positive and the negative training examples with a "gap" (geometric margin).

For now, we will assume that we are given a training set that is linearly separable; i.e., that it is possible to separate the positive and negative examples using some separating hyperplane. How could we find the one that achieves the maximum geometric margin? We will pose the following optimization problem: maximize the margin $\rho = 1/||w||$, such that all points are no closer (on either side) than $|w^T x + b| = 1$ to the separating plane given by $w^T x + b = 0$; thus the constraints reflect our reference choice for scale. Since the labels are the same as the 1,-1 sides of the plane, we can rewrite the constraints as $y(w^T x + b) \geq 1$ for all training points x with label $y \in \{-1, 1\}$ (will have one constraint for each training point).

To make the math nicer we write the objective in terms of $||w||^2$, and we get the following optimization problem:

SVM-PRIMAL OPTIMIZATION PROBLEM

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

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

We've now transformed the problem into a form that can be efficiently solved. The above is an optimization problem with a **convex quadratic objective (1)** and only **linear constraints (2)**. Its solution gives us the optimal margin classifier. This optimization problem can be solved using commercial quadratic programming(QP) code[6] or (better) with duality formulation.

We will use Lagrange duality to solve the above constrained convex optimization problem. This will allow the use kernels, and it is also more efficient.

4 Solution part 1: the dual problem

In this section, we need apply the duality mentioned above to transform the original problems to a easier problem, which can be solved by SMO algorithm, talk about it later. First, we derive the process from original problem to dual problem on separable case and later we will work on the non-separable case. We will show how the dual problem is written in Lagrangian variables α, β, and that w, b are a function of these dual variables (and the data).

4.1 Linearly Separable case

The separable case means the training dataset can be separated by one line, which is shown in Figure 4. We will starts from the original problem:

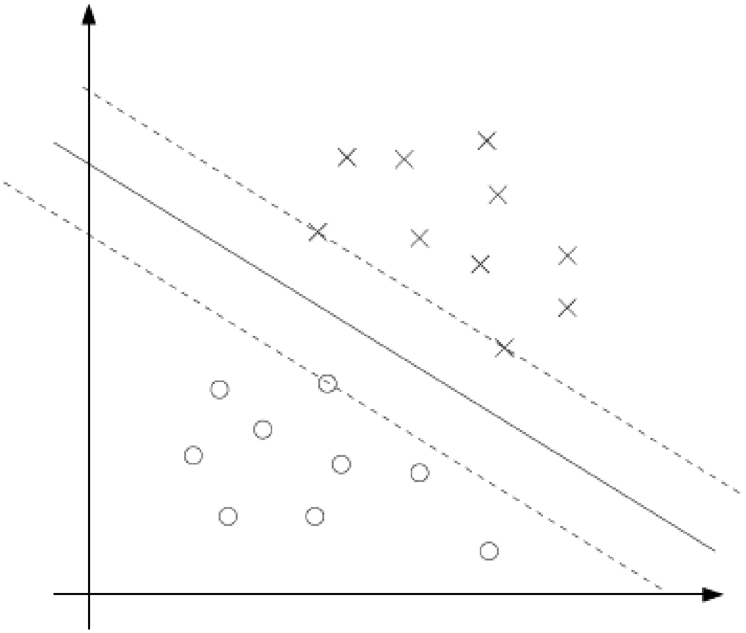

Figure 4: Seperable Example

[6] http://en.wikipedia.org/wiki/Quadratic_programming

SVM-PRIMAL problem

$$\min_{w,b} \frac{1}{2}||w||^2$$
$$\text{s.t.} \quad y_i(w^T x_i + b) \geq 1, \quad i = 1, \dots, m$$

We will first transform the constraints to standard form, and write down the Lagrangian including all constraints

Constraint transformed: $g_i(w, b) = -y_i(w^T x_i + b) + 1 \leq 0$

Lagrangian:

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

Differentiate $\mathcal{L}$ with respect to w, b , and set the differential to zero:

- For w:

$$\frac{\partial}{\partial w}\mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^m \alpha_i y_i x_i = 0$$
$$\implies w = \sum_{i=1}^m \alpha_i y_i x_i \tag{3}$$

- For b:

$$\frac{\partial}{\partial b}\mathcal{L}(w, b, \alpha) = 0 - \sum_{i=1}^m \alpha_i y_i$$
$$\implies \sum_{i=1}^m \alpha_i y_i = 0 \tag{4}$$

Rewrite the Lagrangian objective. Lets put these results back into $\mathcal{L}$ equation in order to eliminate w, b:

$$\begin{aligned}\mathcal{L}(w, b, \alpha) &= \frac{1}{2}||w||^2 - \sum_{i=1}^m \alpha_i(y_i(w^T x_i + b) - 1) \\ &= \frac{1}{2}w^T w - \sum_{i=1}^m \alpha_i y_i w^T x_i - \sum_{i=1}^m \alpha_i y_i b + \sum_{i=1}^m \alpha_i \\ &= \frac{1}{2}\sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^m \alpha_i y_i b + \sum_{i=1}^m \alpha_i \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j x_i^T x_j \end{aligned} \tag{5}$$

We have obtained the Lagrange dual problem for the original SVM-PRIMAL problem. The new variables α, one per datapoint represent the "force" each point pushes the pane away. The equation stated above $\sum_{i=1}^m \alpha_i y_i = 0$ simply states that the plane is in equilibrium as the total force on each side is the same.

It is important to understand the nature of this Lagrangian function: if the linear constraints were equality constraints, typically we'd use the constraints to solve for α-s. But in this case they are inequality constraints (standardized to ≤ 0), which means we cannot simply solve for α by differentiating on α. The KKT theorem (later section) applies to our case (convex objective, linear constraints) and governs the duality with the following rules called **KKT conditions**:

1. the solution for **minimizing** $\mathcal{L}(w, b, \alpha)$ w.r.t. w, b and subject to $\alpha \geq 0$ is the same as the solution of **maximizing** $\mathcal{L}(w, b, \alpha)$ w.r.t. α subject to appropriate constraints.

2. the Lagrangian multipliers are not negative.

3. at solution point, the differential of $\mathcal{L}(w, b, \alpha)$ w.r.t w is zero

4. for equality constraints: at solution point, the differential of $\mathcal{L}(w, b, \alpha)$ w.r.t the Lagrangian multiplier is zero, which is same as saying the constraint is satisfied (we dont have equality constraints here, but we will have them when we introduce slack variables).

5. for inequality constraints: at solution point, either the Lagrangian multiplier is zero and the constraint is satisfied loosely, or multiplier is nonzero and the constrained is satisfied with equality.

The last KKT condition is that for each point $\alpha_i(y_i(w^T x_i + b) - 1) = 0$, or that either $\alpha_i = 0$ or $y_i(w^T x_i + b) = 1$. Thus there are two kinds of training points:

- **support vectors** points for which $\alpha > 0$. These points have an active constraint $y_i(w^T x_i + b) = 1$ which contributes to the equilibrium of the plane and it is **satisfied with equality as the point is on the margin line**. If this point is erased from the training set, the plane will move (equilibrium is changed).

- **non-support vectors** points for which $\alpha = 0$. Such points have a nonactive constraint, which does not contribute to the plane, the constraint is satisfied loosely (perhaps strictly $y_i(w^T x_i + b) > 1$). If this point is erased from the training set, the plane will not move (equilibrium is in the same position).

We will name that last expression of the lagrangian $\mathcal{L}(w, b, \alpha)$, as a function only of α-s, $W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j x_i^T x_j$.

SVM-DUAL OPTIMIZATION PROBLEM

$$\max_\alpha W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j x_i^T x_j \tag{6}$$
$$\text{s.t.} \quad \alpha_i \geq 0, \quad , i = 1, \ldots, m$$
$$\sum_{i=1}^m \alpha_i y^{(i)} = 0.$$

Recover w, b from α-s. Assuming we have solved the dual problem (next section) and we have the solution on α, let's call it α^*. Then we can calculate the solution in original parameters, call it w^*, b^* as following:

$$w^* = \sum_{i=1}^m \alpha^* y_i x_i$$

And as shown in Figure 5, we can first calculate b_A and b_B, then get b^*:

$b_A = \max_{i:y_i=-1} w^{*T} x_i$ This is the maximum on negative points that b^* has to compensate to -1: $b^* \leq -1 - b_A$

$b_B = \min_{i:y_i=1} w^{*T} x_i$ This is the minimum on positive points that b^* has to compensate to 1: $b^* \geq 1 - b_B$

So $1 - b_B \leq b^* \leq -1 - b_A$. We will take b^* to be the average of these two values:

$b^* = \frac{1 - b_B - 1 - b_A}{2} = -\frac{b_A + b_B}{2}$

5 Kernel trick

Will be discussing separately kernels, but for now it is worth to point out the kernel trick: We notice that there are many dot products $(x_i^T x_j)$ in our formula. We can keep the whole SVM-DUAL setup, and the algorithms for solving these problems, but choose **kernel function** $k(x_i^T, x_j)$ to replace the dot products $(x_i^T x_j)$. To

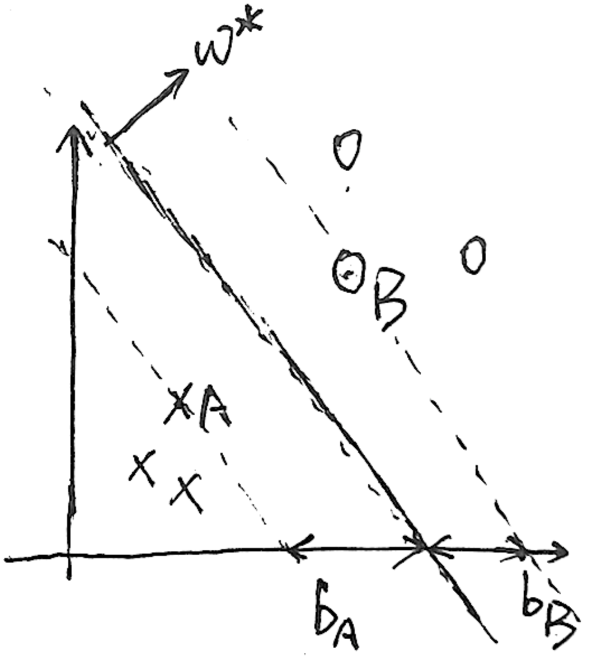

Figure 5: Intercept illustration for b calculation: calculate the b_A, b_B the closest "b" to the plane from either side, then infer b from these tow values.

qualify as a kernel, informally, the function $k(x_i, x_j)$ must be a dot product $k(x_i, x_j) = \Phi(x_i) * \Phi(x_j)$, where $\Phi(x)$ is a mapping vector from the original feature space $\{X\}$ into a different feature space $\{\Phi(X)\}$.

The essential "trick" is that usually Φ is not needed or known, only $k(x_i, x_j)$ is computable and used. To see this for the SVM, it is clear that the dual problem is an optimization written in terms of the dot products replaceable with a given kernel $k(x_i, x_j)$.

How about testing? The parameter $w = \sum_{i=1}^{m} \alpha_i y_i \Phi(x_i)$ is not directly computable if we dont know explicitly the mapping $\Phi()$, but it turns out we dot need to compute w explicitly; we only need to compute predictions for test points z:

$w\Phi(z) + b = \sum_{i=1}^{m} \alpha_i y_i \Phi(x_i)\Phi(z) + b = \sum_{i=1}^{m} \alpha_i y_i k(x_i, z) + b$

This fact has profound implications to the ability of representing data and learning from data: we can apply SVM to separate data which is not linearly separable! Thats because even if the data is not separable in the original space $\{X\}$, it might be separable in the mapped space $\{\Phi(X)\}$. The kernel trick is not specific to SVMs; it works with all algorithms that can be written in terms of dot products $x_i * x_j$.

5.1 Non-Separable case and slack variables

The derivation of the SVM as presented so far assumed that the data is linearly separable. In some cases, it is not clear that finding a separating hyperplane is exactly what we'd want to do, since that might be susceptible to outliers. For instance, the Figure 6, it causes the decision boundary to make a dramatic swing, and the resulting classifier has a much smaller margin.

To make the algorithm work for non-linearly separable datasets as well as be less sensitive to outliers, we

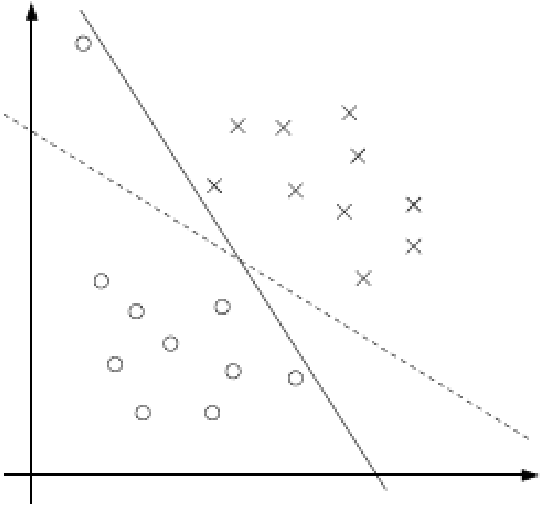

Figure 6: Outlier Example

reformulate the optimization(using $\mathcal{L}_1 \quad regularization$) as following:

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

Thus, examples are now permitted to have margin less than 1, and if an example has functional margin $1 - \xi_i$ (with $\xi > 0$), 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 and of ensuring that most examples have functional margin at least 1.

SVM-DUAL FORM with SLACK VARIABLES

$$\begin{aligned} \max_{\alpha} W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ \text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad , i = 1, \ldots, m \\ \sum_{i=1}^{m}\alpha_i y(i) = 0. \end{aligned} \tag{7}$$

In adding $\mathcal{L}_1 \quad regularization$, the only change to the dual problem is that was originally a constraint that $0 \leq \alpha_i$ has now become $0 \leq \alpha_i \leq C$. The calculation for w^* is the same way, but the calculation for b^* has to be modified (b^* calculation discussed as part of SMO solver). In this case there are three types of training points (see figure 5.1):

- $\alpha = 0$: non interesting points
- $C > \alpha > 0; \beta = 0$: a support vector on the margin line, no slack variable; $\quad y_i(w^T x_i + b) = 1,\ \xi_i = 0$
- $\alpha = C; \beta > 0$: a support vector, inside the side (or even misclassified): $\xi_i > 0$; $y_i(w^T x_i + b) < 1,\ \xi_i > 0$

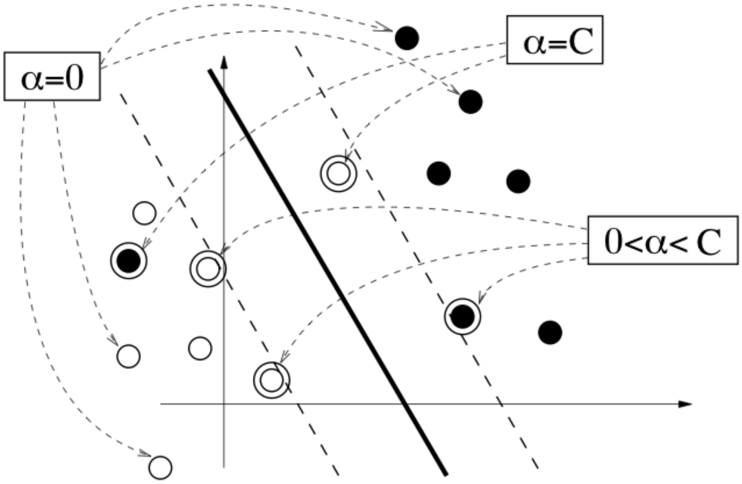

Figure 1.11 A linear function learned by an SVM to discriminate black from white circles, and the corresponding Lagrange multipliers $\boldsymbol{\alpha}$. Each point correctly classified with large confidence ($yf(\mathbf{x}) > 1$) has a null multiplier. Other points are called support vector. They can be on the boundary, in which case the multiplier satisfies $0 \leq \alpha \leq C$, or on the wrong side of this boundary, in which case $\alpha = C$.

5.1.1 Slack variables dual form derivation [optional material]

Let's derive this non-separable problem like we did before. We will have additional constrains for slack variables $\xi_i \geq 0$

1. Non-separable problem

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

2. Constraint transformed:

$$\begin{aligned} g_i(w,b) = 1 - \xi_i - y_i(w^T x_i + b) \leq 0 \\ h_i(w,b) = -\xi_i \leq 0 \end{aligned}$$

3. Lagrangian:

$$\mathcal{L}(w,b,\xi,\alpha,r) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i(y_i(w^T x_i + b) + \xi_i - 1) - \sum_{i=1}^{m} r_i \xi_i$$

4. Set $\theta_{\mathcal{D}}(\alpha, r) = \min_{w,b} \mathcal{L}(w,b,\xi,\alpha,r)$
   Derivate $\mathcal{L}$ with respect to w, b, ξ to zero:

- For w:

$$\begin{aligned}
\frac{\partial}{\partial w}\mathcal{L}(w,b,\xi,\alpha,r) &= w - \sum_{i=1}^{m}\alpha_i y_i x_i = 0 \\
\implies w &= \sum_{i=1}^{m}\alpha_i y_i x_i \qquad (8)
\end{aligned}$$

- For b:

$$\begin{aligned}
\frac{\partial}{\partial b}\mathcal{L}(w,b,\xi,\alpha,r) &= 0 - \sum_{i=1}^{m}\alpha_i y_i = 0 \\
\implies \sum_{i=1}^{m}\alpha_i y_i &= 0 \qquad (9)
\end{aligned}$$

- For ξ:

$$\begin{aligned}
\frac{\partial}{\partial \xi_i}\mathcal{L}(w,b,\xi,\alpha,r) &= C - \alpha_i - r_i = 0 \\
\implies C &= \alpha_i + r_i \ \ \forall i \in \{1,\cdots,m\} \qquad (10)
\end{aligned}$$

5. Put the last three equalities back into $\mathcal{L}$, allows for an objective like before only in lagrangian variables α:

$$\begin{aligned}
\mathcal{L}(w,b,\alpha) &= \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i(y_i(w^Tx_i+b)+\xi_i-1) - \sum_{i=1}^{m}r_i\xi_i \\
&= \frac{1}{2}||w||^2 + \sum_{i=1}^{m}\alpha_i\xi_i + \sum_{i=1}^{m}r_i\xi_i - \sum_{i=1}^{m}\alpha_i(y_i(w^Tx_i+b)-1) - \sum_{i=1}^{m}\alpha_i\xi_i - \sum_{i=1}^{m}r_i\xi_i \\
&= \frac{1}{2}||w||^2 - \sum_{i=1}^{m}\alpha_i(y_i(w^Tx_i+b)-1) \\
&= \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j - \sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{m}\alpha_i y_i b + \sum_{i=1}^{m}\alpha_i \\
&= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j x_i^T x_j \qquad (11)
\end{aligned}$$

Now we get (15), which is the same with (10) in previous derivatives. Although we added more parameters, we only have α now.

6 Solution part 2 : SMO Algorithm instead of Quadratic Solvers

Given the final dual form problem, our goal is to optimize the objective function by given some constraints. First let's see a simple one, which is solving unconstrained optimization problem.

6.1 Coordinate Ascent

If our goal is just to solve an unconstrained optimization problem:

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

The W here is just some function of the parameters α's. To solve this optimization, the idea is that we only choose one parameter, let's say $\hat{\alpha}_i$, and hold all variables α's except α_i. So we can only optimize W with respect to just the parameter $\hat{\alpha}_i$. The algorithms is shown as following:

```
Loop until convergence: {
    For  $i = 1, \ldots, m$, {
        $\alpha_i := \arg\max_{\hat{\alpha}_i} W(\alpha_1, \ldots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \ldots, \alpha_m)$
    }
}
```

And here is an example of coordinate ascent in action: Notice that in Coordinate ascent each step, it only takes a step that's parallel to one of the axes, since only one variable is being optimized at a time.

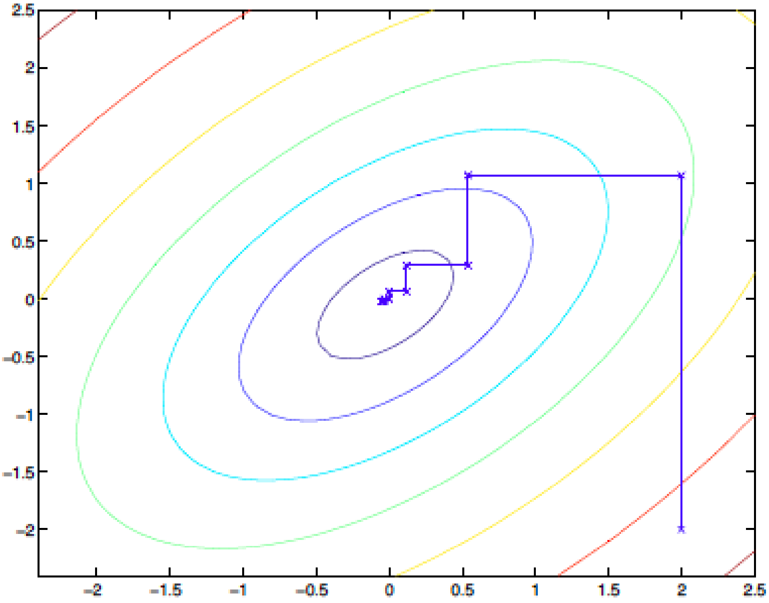

Figure 7: Coordinate Ascent

6.2 SMO

Our dual optimization problem is:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j x_i^T x_j \tag{12}$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad , i = 1, \ldots, m \tag{13}$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0. \tag{14}$$

We want to solve this optimization problem and also satisfy the constraints (18-19). If we still choose one parameter, let's say α_1, and due to the constraint (19), we get $\alpha_1 = -y_i \sum_2^m \alpha_i y_i$, which doesn't work, because the α_1 is also fixed. So how about we choose two parameters instead of just one? This method turns out to be the basic idea behind SMO:

```
Repeat until convergence :{
    select two parameters: $\alpha_i, \alpha_{j:j \neq i}$
    Optimize $W(\alpha)$ with respect to $\alpha_i, \alpha_j$, holding other parameters fixed.
}
```

To know more about the SMO, you can refer the paper written by Platt[7] or the notes from CS 229 in Stanford University[8].

6.3 SMO Pseudocode

Here is a pseudo code, which is exactly following above derivation steps, and it works in practice.

For the pseudo code, there are two main tasks: In outer loop, we choose α_i and α_j and in the inner loop, we update the parameters.

```
Main loop :
  $numChanged = 0$
  $examineAll = True$
  while (numChanged > 0 or examineAll is $True$){
    if (examineAll)
      for all $i$ in training examples
        $numChanged+ = examineExample(i)$
    else
      for all $i$ where $0 < \alpha_i < C$
        $numChanged+ = examineExample(i)$
    if (examineAll)
      $examineAll = False$
    else if (numChanged = 0)
      $examineAll = True$
```

[7] Platt, John (1998), Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines

[8] http://cs229.stanford.edu/materials/smo.pdf

```
examineExamples(i):
  get $y_i, a_i, E_i$
  $r_i = E_i \times y_i$
  if $(r_i < -tol\&a_i < C)||(r_i > tol\&a_i > 0)$
  {
    if number of non-boundaries points$(0 < \alpha < C) > 1$:
    {
      $j = argmax_j|Ei - Ej|$
      if takeStep(i,j):
        return 1
    }
    loop all non-boundary points randomly:
    {
      $j =$ random(non-boundary points)
      if takeStep(i,j):
        return 1
    }
    loop all points randomly:
    {
      $j =$ random(Entire points)
      if takeStep(i,j):
        return 1
    }
  }
  return 0
```

```
takeStep(i, j)
  if $i == j$:
    return false
  get $y_j, a_j, E_j$
  $s = y_i * y_j$
  $if\ y_i \neq y_j : L = \max(0, \alpha_j - \alpha_i)$ and $H = \min(C, \alpha_j - \alpha_i + C)$
  $if\ y_i = y_j : L = \max(0, \alpha_i + \alpha_j - C)$ and $H = \min(C, \alpha_i + \alpha_j)$
  if $L == H$:
    return false
  $K_{i,i} = kernel(i, i)$
  $K_{j,j} = kernel(j, j)$
  $K_{i,j} = kernel(i, j)$
  $\eta = K_{i,i} + K_{j,j} - 2K_{i,j}$
  if $\eta \leq 0$:
    return false
  $\alpha_j^{new} = \alpha_j + \frac{y_j(E_i - E_j)}{\eta}$
  $\alpha_j^{new,clipped} = \begin{cases} L & \text{if } \alpha_j^{new} < L \\ \alpha_j^{new} & \text{if } \alpha_j^{new} \in [L, H] \\ H & \text{if } \alpha_j^{new} > H \end{cases}$
  if $|\alpha_j - \alpha_j^{new,chipped}| < \epsilon(\alpha_j + \alpha_j^{new,chipped} + \epsilon)$:
    return false
  $\alpha_i^{new} = \alpha_i + s(\alpha_j - \alpha_j^{new,chipped})$
  Update $b^{new}$, See end of section 6.
  return true
```

6.4 SMO Details

Since we just mentioned that in SMO, there are two main steps: 1) select two parameters, α_i, and α_j; 2) optimize $W(\alpha)$ with respect to α_i, α_j, holding other parameters fixed. In this section, we will give more details about how to deal with these two steps.

6.4.1 How to Select α_i and α_j

The basic idea is that we want to choose the worst α_i and α_j in each step to modify/correct them to make the biggest progress towards the global maximum. For example in Figure 8, we definitely will choose P_1 as the path from point A to G.

So what's the worst α? Let's first choose α_i and then α_j.

- **Selecting α_i**
  The idea about choosing the worst α_i is based on who violates the KKT constrains dual-complementary. Let's recall the KKT dual-complementary(A, B, C, D, E see in Figure 9):

$$KKT\ dual\ complementary : \begin{cases} \alpha_i = 0 \implies y_i f(x_i) \geq 1 & (\xi = 0 \quad \text{correct points, like C,D,E}) \\ \alpha_i = C \implies y_i f(x_i) \leq 1 & (\xi \geq 0 \quad \text{mistake, like B}) \\ \alpha_i \in (0, C) \implies y_i f(x_i) = 1 & (\xi = 0 \quad \text{support vector, like A}) \end{cases}$$

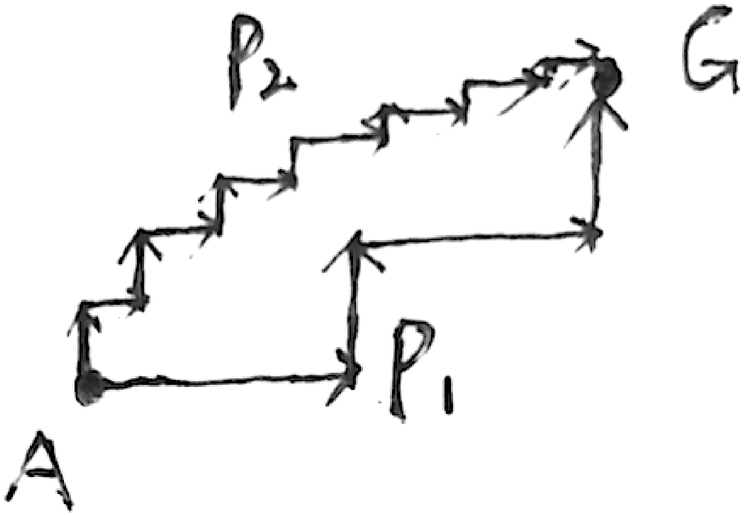

Figure 8: Idea about Choosing α

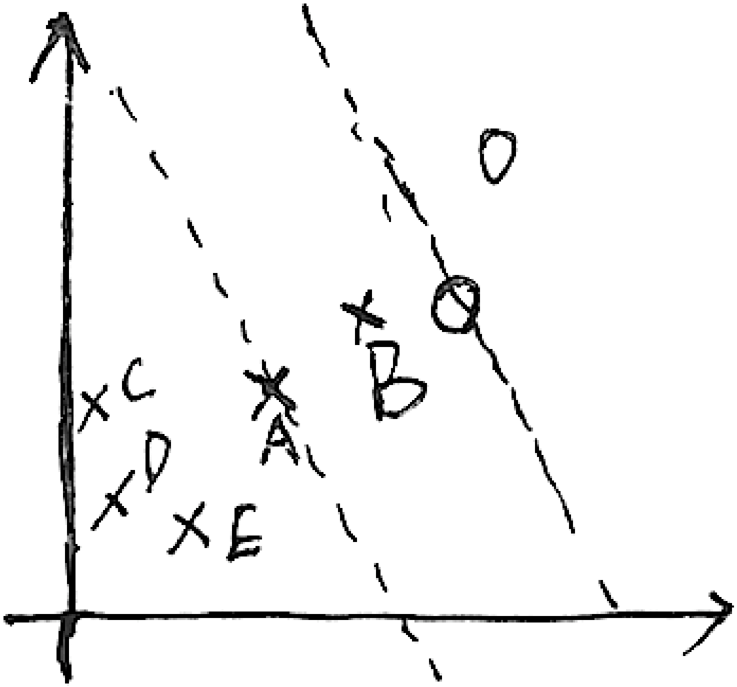

Figure 9: KKT dual complementary α

where $f(x_i) = w^T x_i + b$. Thus, violations of KKT is shown as following:

$$Violations\ of\ KKT \begin{cases} \alpha_i = 0 \ \&\& \ y_i f(x_i) < 1 \\ \alpha_i = C \ \&\& \ y_i f(x_i) > 1 \\ \alpha_i \in (0, C) \ \&\& \ y_i f(x_i) \neq 1 \end{cases}$$

Let's talk more about the violations of KKT. Why does it violate the KKT? We start from introducing the gap between primal problem and dual problem. We define the gap for a point (x_i, y_i) is:

$$\begin{aligned}Gap_i &= \alpha_i(y_i(\sum_{j=1}^{m} \alpha_j y_j < x_j, x_i >) - 1) + C\xi_i \\ &= \alpha_i(y_i f(x_i) - 1 - y_i b) + C\xi_i\end{aligned}$$

And we also define:

$$\xi_i = \max(0, 1 - y_i f(x_i))$$

Let's see each violation of KKT:

1.
$$\begin{aligned}&Satisfy: \\ &\qquad \alpha_i = 0\&\&y_i f(x_i) \geq 1, \xi = 0 \implies Gap_i^s = 0 \\ &Violate: \\ &\qquad \alpha_i = 0\&\&y_i f(x_i) < 1, \xi > 0 \implies Gap_i^v = C\xi_i > Gap_i^s\end{aligned}$$

2.
$$\begin{aligned}&Satisfy: \\ &\qquad \alpha_i = C\&\&y_i f(x_i) \leq 1, \xi = 1 - y_i f(x_i) \implies \\ &\qquad Gap_i^s = C(y_i f(x_i) - 1 - y_i b) + C(1 - y_i f(x_i))0 = -Cy_i b \\ &Violate: \\ &\qquad \alpha_i = C\&\&y_i f(x_i) > 1, \xi = 0 \implies \\ &\qquad Gap_i^v = C(y_i f(x_i) - 1) - Cy_i b > Gap_i^s\end{aligned}$$

3. Easily we can also prove that $Gap_i^v > Gap_i^s$ here.

We notice that the gap between primal problem and dual problem will be increased if violating the KKT dual complementary. Thus, we want to choose α_i, who violate the KKT, and modify it to reduce the gap.

Further, we know that there are three violations, which one should we deal first? In fact, in the algorithm, we want to choose all the samples with $\alpha_i \in (0, C)$ first, which are the support vectors. Why? Because non-bound samples have more probabilities to be modified, which means the support vectors effect the hyperplane more than the points, who are not on the hyperplane. After we can not find any non-bound samples, who violate KKT, then we go through the whole samples to check if violating other two KKT dual complementary.

- **Selecting** α_j Select the α_j will follow the rule:

$$\max_{\alpha_j} |E_i - E_j|$$

where the $E_i = f(x_i) - y_i$, we will explain why during the later derivation.

6.4.2 How to Optimize $W(\alpha)$ respect to α_i and α_j

After choosing the α_i and α_j, now we would talk about how to optimize the $W(\alpha)$. To make it easy to explain, we set $\alpha_1 = \alpha_i$ and $\alpha_2 = \alpha_j$, we also define:

$$w = \sum_{i=1}^{m} y_i \alpha_i x_i \tag{15}$$

$$f(x_i) = w^T x_i + b \tag{16}$$

$$K_{i,j} =< x_i, x_j > \quad \text{where using kernel trick} \tag{17}$$

$$v_i = \sum_{j=3}^{m} y_j \alpha_j K_{i,j} = f(x_i) - \sum_{j=1}^{2} y_j \alpha_j K_{i,j} - b \tag{18}$$

$$E_i = f(x_i) - y_i \tag{19}$$

$$\eta = K_{i,i} + K_{j,j} - 2K_{i,j} = ||\phi(x_i) - \phi(x_j)||^2 \tag{20}$$

1. **Calculate α_j**
   First, put α_1 and α_2 to objective function:

$$\begin{aligned}
W(\alpha) &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j x_i^T x_j \\
&= \alpha_1 + \alpha_2 + \sum_{i=3}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} (\sum_{j=1}^{2} y_i y_j \alpha_i \alpha_j K_{i,j} + \sum_{j=3}^{m} y_i y_j \alpha_i \alpha_j K_{i,j}) \\
&= \alpha_1 + \alpha_2 + \sum_{i=3}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{2} (\sum_{j=1}^{2} y_i y_j \alpha_i \alpha_j K_{i,j} + \sum_{j=3}^{m} y_i y_j \alpha_i \alpha_j K_{i,j}) \\
&\qquad - \frac{1}{2} \sum_{i=2}^{m} (\sum_{j=1}^{2} y_i y_j \alpha_i \alpha_j K_{i,j} + \sum_{j=3}^{m} y_i y_j \alpha_i \alpha_j K_{i,j}) \\
&= \alpha_1 + \alpha_2 + \sum_{i=3}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{2} \sum_{j=1}^{2} y_i y_j \alpha_i \alpha_j K_{i,j} - \frac{1}{2} \sum_{i=1}^{2} \sum_{j=3}^{m} y_i y_j \alpha_i \alpha_j K_{i,j} \\
&\qquad - \frac{1}{2} \sum_{i=3}^{m} \sum_{j=1}^{2} y_i y_j \alpha_i \alpha_j K_{i,j} - \frac{1}{2} \sum_{i=3}^{m} \sum_{j=3}^{m} y_i y_j \alpha_i \alpha_j K_{i,j} \\
&= \alpha_1 + \alpha_2 + \sum_{i=3}^{m} \alpha_i - \frac{1}{2} y_1^2 \alpha_1^2 K_{1,1} - \frac{1}{2} y_2^2 \alpha_2^2 K_{2,2} - y_1 y_2 \alpha_1 \alpha_2 K_{1,2} \\
&\qquad - y_1 \alpha_1 \sum_{j=3}^{m} y_j \alpha_j K_{1,j} - y_2 \alpha_2 \sum_{j=3}^{m} y_j \alpha_j K_{2,j} - \frac{1}{2} \sum_{i=3}^{m} \sum_{j=3}^{m} y_i y_j \alpha_i \alpha_j K_{i,j} \\
&= \alpha_1 + \alpha_2 - \frac{1}{2} K_{1,1} \alpha_1^2 - \frac{1}{2} K_{2,2} \alpha_2^2 - y_1 y_2 K_{1,2} \alpha_1 \alpha_2 - y_1 \alpha_1 v_1 - y_2 \alpha_2 v_2 + CONSTANT
\end{aligned}$$

   In fact, now we can apply two constrains (18) and (19) to solve this problem.
   Constrain (19): Due to $\sum_{i=1}^{m} \alpha_i y_i = 0$, and $\alpha_3, \ldots, \alpha_m$ and $y_3, \ldots, y_m$ are fixed, so we can set:

$$\alpha_1 y_1 + \alpha_2 y_2 = C'$$

and we can get

$$\begin{aligned}
&y_1(\alpha_1 y_1 + \alpha_2 y_2) = y_1 C' \\
\Longrightarrow\ &\alpha_1 + \alpha_2 y_1 y_2 = y_1 C' \\
\Longrightarrow\ &\alpha_1 = y_1 C' - \alpha_2 y_1 y_2 \quad \text{Set } y_1 C' = \gamma \text{ and } y_1 y_2 = s \\
\Longrightarrow\ &\alpha_1 = \gamma - \alpha_2 s
\end{aligned}$$

Now put α_1 to $W(\alpha)$:

$$\begin{aligned}
W(\alpha) = \gamma - s\alpha_2 + \alpha_2 - \frac{1}{2}K_{1,1}(\gamma - s\alpha_2)^2 - \frac{1}{2}K_{2,2}\alpha_2^2 - sK_{1,2}(\gamma - s\alpha_2)\alpha_2 \\
- y_1(\gamma - s\alpha_2)v_1 - y_2\alpha_2 v_2 + CONSTANT
\end{aligned}$$

And derivate $W(\alpha)$ respect to α_2:

$$\begin{aligned}
\frac{\partial W(\alpha)}{\partial \alpha_2} &= -s + 1 + K_{1,1}s\gamma - K_{1,1}\alpha_2 - K_{2,2}\alpha_2 - s\gamma K_{1,2} + 2K_{1,2}\alpha_2 + y_2 v_1 - y_2 v_2 \\
&= 0
\end{aligned}$$

Put $s = y_1 y_2$ and $y_2^2 = 1$, we can get:

$$\begin{aligned}
&y_2(y_2 - y_1 + y_1\gamma(K_{1,1} - K_{1,2}) + v_1 - v_2) - \alpha_2(K_{1,1} + K_{2,2} - 2K_{1,2}) = 0 \\
\Longrightarrow \alpha_2^{new} = &\frac{y_2(y_2 - y_1 + y_1\gamma(K_{1,1} - K_{1,2}) + v_1 - v_2)}{K_{1,1} + K_{2,2} - 2K_{1,2}}
\end{aligned}$$

After that, we combine$\gamma = \alpha_1^{old} + s\alpha_2^{old}$, (24)and (25) and get:

$$\begin{aligned}
\alpha_2^{new} &= \alpha_2^{old} + \frac{y_2(E_1 - E_2)}{\eta} \\
\Longrightarrow \alpha_j^{new} &= \alpha_j^{old} + \frac{y_j(E_i - E_j)}{\eta}
\end{aligned} \tag{21}$$

Constrain (18): Due to $0 \leq \alpha_i \leq C$, so the α_2 and α_1 must be into a $(0, C) \times (0, C)$ box, shown in the Figure 10. And as defined in previous: $\alpha_1 y_1 + \alpha_2 y_2 = C'$. Now we can consider that there are two different situations:

- **if y_1, y_2 are the same value:** $[y_1 y_2 = 1]$
  Then we can get $\alpha_1 + \alpha_2 = C'(or - C'$which is the same). For example in the Figure 11, we could get

  $$\begin{aligned}
  &\alpha_2 \in [0, C] \ \& \ \alpha_2 \in [C' - C, C'] \\
  \Longrightarrow\ &\alpha_2 \in [0, C] \ \& \ \alpha_2 \in [\alpha_1 + \alpha_2 - C, \alpha_1 + \alpha_2]
  \end{aligned}$$

  Combine the lower bound and upper bound we could get:

  $$\begin{aligned}
  L_{\alpha_2} &= \max(0, \alpha_1 + \alpha_2 - C) \\
  H_{\alpha_2} &= \min(C, \alpha_1 + \alpha_2)
  \end{aligned}$$

- **if y_1, y_2 are different value:** $[y_1 y_2 = -1]$
  Then we could get $\alpha_1 - \alpha_2 = C'(or - C'$which is the same). For example in the Figure 12, we could get

  $$\begin{aligned}
  &\alpha_2 \in [0, C] \ \& \ \alpha_2 \in [-C', C - C'] \\
  \Longrightarrow\ &\alpha_2 \in [0, C] \ \& \ \alpha_2 \in [\alpha_2 - \alpha_1, \alpha_2 - \alpha_1 + C]
  \end{aligned}$$

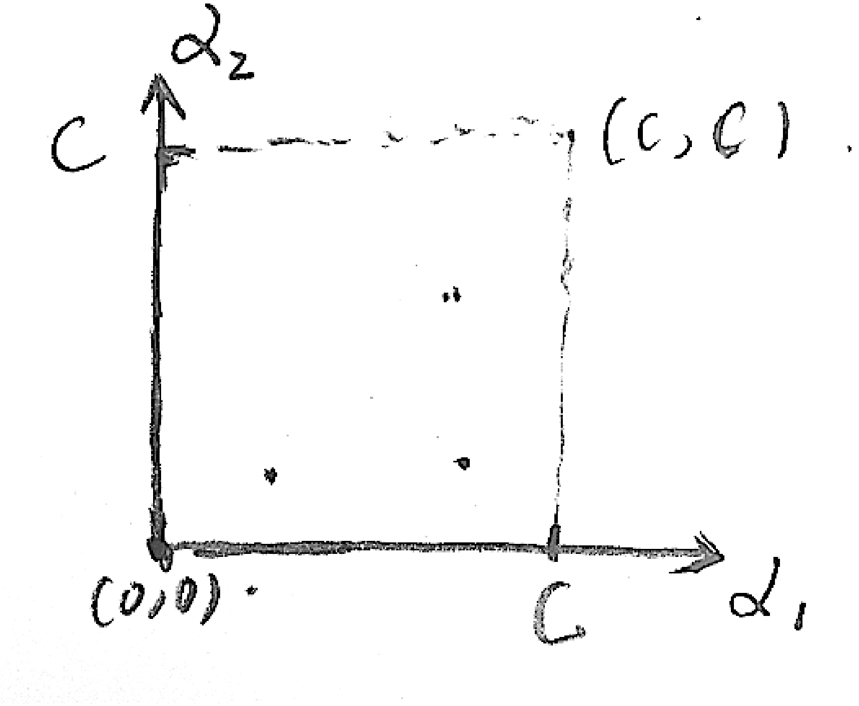

Figure 10: Constrains on α

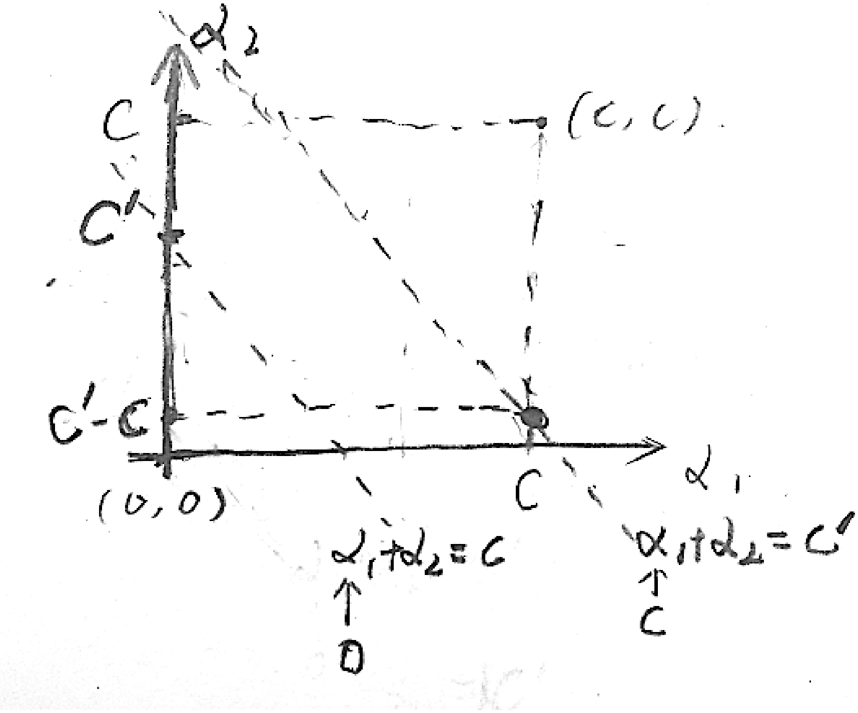

Figure 11: y_1, y_2 same value

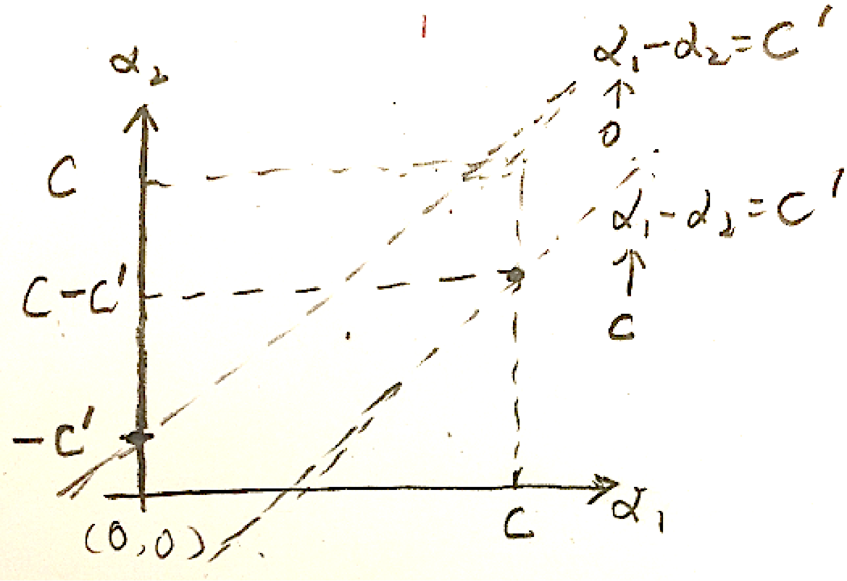

Figure 12: y_1, y_2 different value

Combine the lower bound and upper bound we could get:

$$\begin{aligned} L_{\alpha_2} &= \max\left(0, \alpha_2 - \alpha_1\right) \\ H_{\alpha_2} &= \min\left(C, \alpha_2 - \alpha_1 + C\right) \end{aligned}$$

Generally:

$$\begin{aligned} &if\ y_i \neq y_j : L = \max\left(0, \alpha_j - \alpha_i\right) \text{ and } H = \min\left(C, \alpha_j - \alpha_i + C\right) \\ &if\ y_i = y_j : L = \max\left(0, \alpha_i + \alpha_j - C\right) \text{ and } H = \min\left(C, \alpha_1 + \alpha_2\right) \end{aligned}$$

Then we can use this limitations to clip the α_j like:

$$\alpha_j^{new,clipped} = \begin{cases} L & \text{if } \alpha_j^{new} < L \\ \alpha_j^{new} & \text{if } \alpha_j^{new} \in [L, H] \\ H & \text{if } \alpha_j^{new} > H \end{cases}$$

2. **Calculate α_i**
Once we have got new α_j, it won't be hard to calculate α_i, notice that we already have:

$$\begin{aligned} &\alpha_i^{old} = \gamma - s\alpha_j^{old} \\ &\alpha_i^{new} = \gamma - s\alpha_j^{new,clipped} \\ &s = y_i y_j \\ \Longrightarrow\ &\alpha_i^{new} = \alpha_i^{old} + s\alpha_i^{old} - s\alpha_j^{new,clipped} \\ &\qquad = \alpha_i^{old} + y_i y_j(\alpha_j^{old} - \alpha_j^{new,clipped}) \end{aligned}$$

3. **Calculate b**
   When we calculate the bias b, we are based on the point whether it's the non-bound points(support vector) or not. (Like point A in Figure 9):

$$y_i f(x_i) = 1.$$

   Now we can divide the situations to four parts:

   - **if only $\alpha_1^{new} \in (0, C)$:**

$$\begin{aligned} & y_1 f(x_1) = 1 \\ \Longrightarrow\ & y_1(\alpha_1^{new} y_1 K_{1,1} + \alpha_2^{new,clipped} y_2 K_{2,1} + \sum_{i=3}^{m}(\alpha_i y_i K_{i,1}) + b_1^{new}) = 1 \\ & E_1 = f(x_1) - y_1 = \alpha_1^{old} y_1 K_{1,1} + \alpha_2^{old} y_2 K_{2,1} + \sum_{i=3}^{m}(\alpha_i y_i K_{i,1}) + b^{old} - y_1 \\ \Longrightarrow\ & \sum_{i=3}^{m}(\alpha_i y_i K_{i,1}) = E_1 - \alpha_1^{old} y_1 K_{1,1} - \alpha_2^{old} y_2 K_{2,1} - b^{old} + y_1 \end{aligned}$$

     combine them

$$\Longrightarrow b_i^{new} = b_1^{new} = b^{old} - E_1 + (\alpha_1^{old} - \alpha_1^{new}) y_1 K_{1,1} + (\alpha_2^{old} - \alpha_2^{new}) y_2 K_{2,1}$$

   - **if only $\alpha_2^{new} \in (0, C)$:**
     Same thing we could get

$$b_j^{new} = b_2^{new} = b^{old} - E_2 + (\alpha_1^{old} - \alpha_1^{new}) y_1 K_{1,2} + (\alpha_2^{old} - \alpha_2^{new}) y_2 K_{2,2}$$

   - **if both are non-bound:**
     Choose one of b_1^{new} or b_2^{new}.
   - **if both are not non-bound:**
     We choose could set b^{new} as:

$$b^{new} = \frac{b_1^{new} + b_2^{new}}{2}$$

   Generally, we update b as:

$$b^{new} = \begin{cases} b_i^{new} & \text{if } \alpha_i^{new} \in (0, C) \\ b_j^{new} & \text{if } \alpha_j^{new,clipped} \in (0, C) \\ \frac{b_i^{new} + b_j^{new}}{2} & \text{otherwise} \end{cases}$$

7 Lagrange Duality [optional material]

As mentioned before, the problem is solving constrained optimization problems. In fact, we are familiar with the problem of optimization without any constrains, we can consider gradient descent, Newtown's method, interval cutting etc. Before solving the constrained optimization problems, we need talk about the Lagrangian, Primal and Dual problems.

7.1 Lagrange

Lagrange multipliers to solve the problem of the following form, whose constrain is an equality:

$$\begin{aligned} & \min_w f(w) \\ \text{s.t.} \quad & h_i(w) = 0, \quad i = 1, \ldots, l \end{aligned}$$

We can use Lagrange multipliers on it. And in this method, we define the Lagrangian as following:

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

We would find and set $\mathcal{L}$'s partial derivatives to zero:

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0; \quad \frac{\partial \mathcal{L}}{\partial \beta_i} = 0$$

And then we can get the w^* to be the solution from the partial derivatives step.

7.2 Primal Problem

Consider the following, which we will call the primal optimization problems, whose has inequality as well as equality constraints.

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

Then we can define the generalized Lagrangian

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

Here consider the quantity

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

In the problem $\theta_{\mathcal{P}}(w)$, if $g_i(w) > 0$ or $f_i(w) \neq 0$, which violates any of the primal constraints given above, then you should be able to verify that

$$\begin{aligned} \theta_{\mathcal{P}}(w) &= \max_{\alpha,\beta:\alpha_i \geq 0} f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w) \\ &= \infty \end{aligned}$$

Conversely, if the constraints are indeed satisfied for a particular value of w, then $\theta_{\mathcal{P}}(w) = f(w)$. Hence,

$$\theta_{\mathcal{P}}(w) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{otherwise} \end{cases}$$

Thus, $\theta_{\mathcal{P}}$ takes the same value as the objective in our problem for all values of w that satisfies the primal constraints, and is positive infinity if the constraints are violated. Hence the minimization problem has been transformed to

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

For later use, we define $p^* = \min_w \theta_{\mathcal{P}}(w)$ as the value of the primal problem. In fact, we see that primal problem has the same solutions as our original problem.

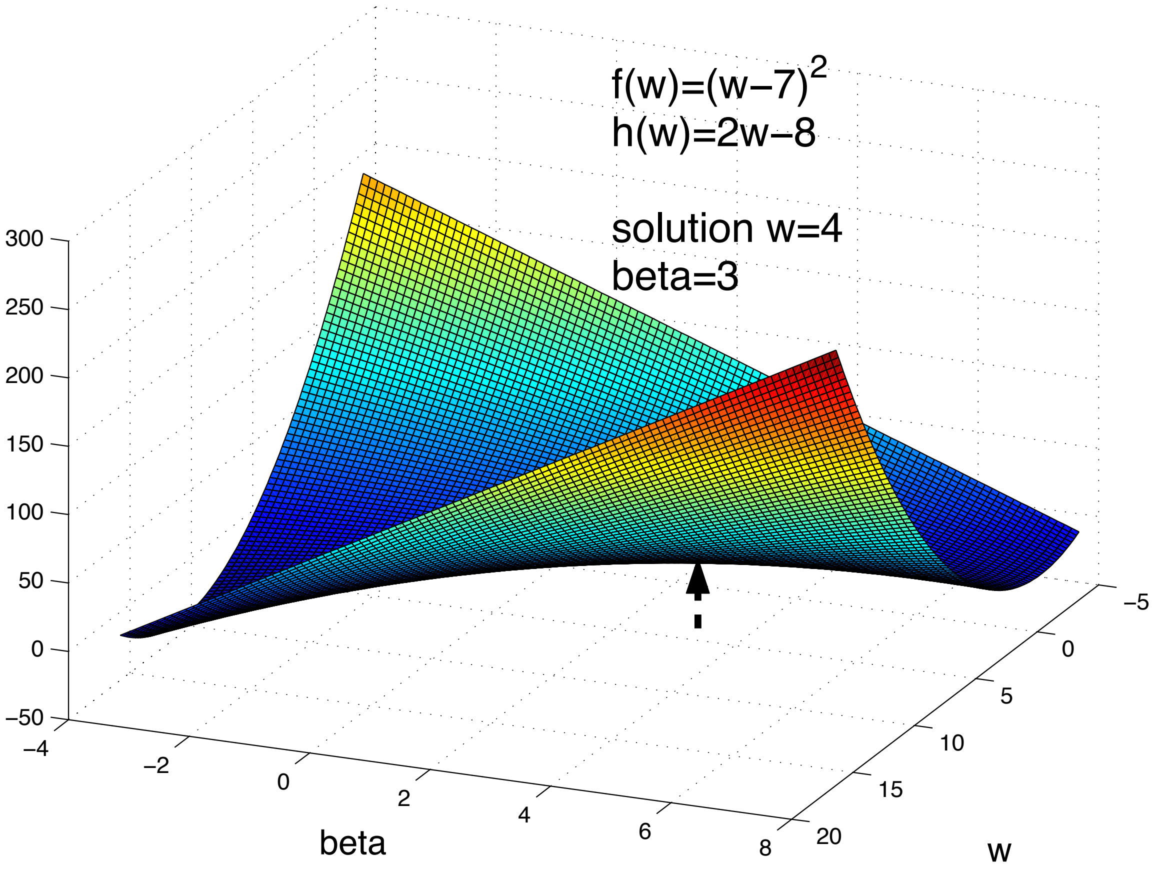

Figure 13: Saddle Point

7.3 Dual Problem

Then we can define

$$\theta_{\mathcal{D}}(\alpha, \beta) = \min_{w} \mathcal{L}(w, \alpha, \beta)$$

and then pose the dual optimization problem:

$$\max_{\alpha,\beta:\alpha\geq 0} \theta_{\mathcal{D}}(\alpha, \beta) = \max_{\alpha,\beta:\alpha\geq 0} \min_{w} \mathcal{L}(w, \alpha, \beta)$$

And we also define $d^* = \max_{\alpha,\beta:\alpha\geq 0} \theta_{\mathcal{D}}(\alpha, \beta)$. We can see that dual problem is pretty similar to our primal problem shown above, except that the order of the "max" and the "min" are now exchanged.

For problem with convex objectives and linear constraints the **duality gap always closes** (KKT theorem) in the sense that

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

The solution is exactly this "saddle point" : maximum of the minimums of each convex slice, same as minimum of the maximums of each concave slice (shown in Figure13.

7.4 Karush-Kuhn-Tucker conditions for duality gap

How are the primal and the dual problems related? And why should we introduce primal and dual problems? We will talk a little bit in this section. Let's start with why. Since we present our original problem as following:

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

By introducing Lagrange multipliers α, the original constrained problem can be expressed as a primal problem:

$$\begin{aligned}w^*, b^* &= \arg p^* = \arg \min_{w,b} \theta_{\mathcal{P}}(w, b)\\ &= \arg \min_{w,b} \max_{\alpha \geq 0} (\frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i (y_i(w^T x_i + b) - 1))\end{aligned}$$

this is a saddle point[9]. If we want to solve this primal problem, we can use QP, which is inefficient. We try to transform the primal problem to the dual problem as following[10]:

$$\begin{aligned}\alpha^* &= \arg d^* = \arg \max_{\alpha} \theta_{\mathcal{D}}(\alpha)\\ &= \arg \max_{\alpha} (\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j)\\ \text{s.t.} \quad &\alpha_i \geq 0\\ &\sum_{i=0}^{m} \alpha_i y_i = 0\end{aligned}$$

In dual problem, we get rid off two parameters w, b and the constrains are much easier than before. BTW, notice that we have $x_i^T x_j$ in the formula, which gives us a chance apply kernel trick on it. We will talk about it later.

We can notice that the dual problem is much better than primal problem. If we can transform the original problem to primal problem, and then to dual problem, it will be good steps to the solutions. In fact, the there is some relationship between primal and dual problems. Notice a fact that $\max \min(f) \leq \min \max(f)$, thus

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

That's $d^* \leq p^*$. Further, $d^* = p^*$ under the KKT conditions. Once the Primal problem and Dual problem equal to each other, the parameters will meet the KKT conditions. We just introduce the five conditions as following:

$$\frac{\partial}{\partial w_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \ldots, n \tag{22}$$

$$\frac{\partial}{\partial \beta_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \ldots, l \tag{23}$$

$$\alpha_i^* g_i(w^*) = 0, \quad i = 1, \ldots, k \tag{24}$$

$$g_i(w^*) \leq 0, \quad i = 1, \ldots, k \tag{25}$$

$$\alpha^* \geq 0, \quad i = 1, \ldots, k \tag{26}$$

Later, we will just apply KKT conditions on primal problem to get the dual form problem.

[9] http://en.wikipedia.org/wiki/Saddle_point

[10] http://en.wikipedia.org/wiki/Support_vector_machine