Linear Models

Karl Hajjar

August 2019

1 The Perceptron

The perceptron is probably the first linear model to be designed for Machine Learning. It dates back to the 50s' and can be seen as the ancestor of neural networks, a.k.a. "Multi-Layer Perceptrons" (MLPs). The perceptron was designed when not much was known about Machine Learning algorithms, and the theory was only nascent. It is therefore a pretty weak linear model by today's standards and suffers from a couple of major flaws, but still has interesting properties which we discuss here.

We consider the binary classification problem with inputs $x \in \mathbb{R}^d$ and targets $y \in \{-1, 1\}$. The perceptron tries to learn a linear classifier as $\text{sgn}(w^T x + b)$. That is, given a data set $\{(x_1, y_1), \dots, (x_n, y_n)\}$, the perceptron tries to find the "best" parameters $(w, b) \in \mathbb{R}^d \times \mathbb{R}$. In order to make things simpler, we can get rid of the bias and put all the parameters in a weight vector $\theta \in \mathbb{R}^{d+1}$ by adding an artificial dimension to the inputs x whose value is always 1. We will thus consider in this section a vector $\theta \in \mathbb{R}^{d+1}$ and inputs x also in $\mathbb{R}^{d+1}$.

What the "best parameters" means in the case of the perceptron is parameters for which as many of the training samples as possible are classified correctly by the model. The perceptron model learns in an *online* fashion, *i.e.* the weight parameters are updated on-the-fly as the model sees the training examples one by one (and possibly loops over the training set). The perceptron starts by setting the weight vector θ_0 to zero, and, at each step $t \geq 0$, observes a new sample x_{i_t} and updates the weight vector using the following update rule:

$$\begin{cases} \theta_{t+1} = \theta_t + y_{i_t} x_{i_t} \text{ if } x_{i_t} \text{ is misclassified } (\textit{i.e. } y_{i_t} \theta_t^T x_{i_t} < 0) \\ \theta_{t+1} = \theta_t \text{ otherwise} \end{cases} \tag{1}$$

This update makes sense because ultimately we want that $y_i \theta^T (x_i) > 0$ for any sample i, and we thus try to enforce that behaviour by adding the vector $y_i x_i$ to the current vector θ as long as sample i gets misclassified. Actually, when taking a closer look at this update rule, we observe that it actually corresponds to computing a gradient step (with learning rate 1) on a single misclassified example using the loss term $-y_i x_i^T \theta^* = |x_i^T \theta^*|$, which is actually the distance (if $||\theta^*||_2 = 1$) of the misclassified point to the hyperplane. In other words, if we take the loss to be 0 if the sample is classified well and the distance to the hyperplane otherwise, than we can see the perceptron as minimizing the empirical loss on the training data using stochastic gradient descent.

Let us now assume that the training data is linearly separable, which is to say there is a $\theta^* \in \mathbb{R}^{d+1}$ such that:

$$\forall i \in [\![1, n]\!],\ y_i(\theta^T x_i) > 0 \tag{2}$$

The main theoretical results with the perceptron is that, under the previous assumption of linear separability, it can be proven that the perceptron is able to find a weight vector that classifies well all the examples in the training set after a finite number of iterations.

In what follows we provide a proof of this result, along with a couple of comments and remarks. First, since the training data set is finite, let us denote:

$$m = \min_{i \in [\![1,n]\!]} y_i(\theta^T x_i) \tag{3}$$

and

$$R = \max_{i \in [\![1,n]\!]} ||x_i||_2 \tag{4}$$

Given the separability assumption (2), we have that $m > 0$. Given the update rule (1), we observe that an update is made to the weight vector only when the perceptron fails to classify correctly the example at hand. Our aim is thus to show that the number of updates of the weight vector is upper bounded. We will do that by first showing that at each update the value of the quantity $\theta_t^T \theta^*$ increases at least by a fixed, (strictly) positive amount. In a second step we will show that this quantity is upper bounded whatever the value of θ_t might be, given the update rule, which will conclude the proof.

Given that a change in the value of θ only happens when there is a misclassification, we can consider the values of θ only at the steps where there is an update, since between two updates this values does not change (but a certain number of examples from the training set might still be shown to the perceptron). Let k be a positive integer ($k \geq 1$). Let us consider what happens when the value of the weight vector changes from θ_{k-1} to θ_k (irrespective of the existence of such a k-th update for now). Let us denote by i_k the index of the training sample that was shown to the perceptron before the update, and that was thus misclassified by it. Given the definition of m (3), we have that:

$$\theta_k^T \theta^* = \theta_{k-1}^T \theta^* + y_{i_k}(x_{i_k}^T \theta^*)$$
$$\theta_k^T \theta^* \geq \theta_{k-1}^T \theta^* + m$$

By induction, and given that $\theta_0 = 0$, this leads to:

$$\theta_k^T \theta^* \geq km \tag{5}$$

On the other hand, we have the following upper bound, using Cauchy-Schwarz's inequality:

$$(\theta_k^T \theta^*)^2 \leq ||\theta_k||_2^2 \times ||\theta^*||_2^2$$

Moreover, using the fact that sample i_k is misclassified by the perceptron before the k-th update, we get:

$$\begin{aligned}
||\theta_k||_2^2 &= ||\theta_{k-1} + y_{i_k} x_{i_k}||_2^2 \\
||\theta_k||_2^2 &= ||\theta_{k-1}||_2^2 + 2\underbrace{y_{i_k} \theta_{k-1}^T x_{i_k}}_{<0} + \underbrace{||x_{i_k}||_2^2}_{\leq R^2 \text{ by } (4)} \\
||\theta_k||_2^2 &\leq ||\theta_{k-1}||_2^2 + R^2
\end{aligned}$$

By induction, and given that $||\theta_0||_2 = 0$, we have that:

$$||\theta_k||_2^2 \leq kR^2$$

This combined to the result of Cauchy-Schwarz's inequality yields:

$$(\theta_k^T \theta^*)^2 \leq k||\theta^*||_2^2 R^2 \tag{6}$$

Combining inequalities (5) and (6), we finally get:

$$km \leq \theta_k^T \theta^* \leq |\theta_k^T \theta^*| \leq \sqrt{k}||\theta^*||_2 R$$

and, since $m > 0$, we have:

$$\sqrt{k} \leq \frac{||\theta^*||_2 R}{m}$$

so that

$$k \leq \left(\frac{||\theta^*||_2 R}{m}\right)^2 \tag{7}$$

The inequality (7) above shows that if there are any updates during the learning of the weight vector with the perceptron, their number is bounded, and there is therefore a time step from which there is no more updates, which means that the learnt perceptron at this time step classifies correctly all the training examples.

Let us finish this section by a couple of remarks. First, if we take θ^* to be of ℓ_2 norm 1, than m is actually the minimal distance from the points in the training set to the hyperplane defined by θ^*, also known as the margin. We see that in this case, the number of updates before the perceptron classifies all examples in the training set is the square of the ratio between the largest distance to the origin in the training set and the margin. This shows the importance of the margin in linear models, which was later widely studied in SVMs, and is tightly linked to the generalisation abilities of such linear models.

Secondly, even though the perceptron is guaranteed to converge in a finite number of updates to a solution that classifies correctly all points in the training set when the training data is separable, there is no such guarantee when the training set is not separable, and in fact it can be shown that in that case the perceptron with the update rule (1) does not converge.

Finally, even in the separable case, if the perceptron is able to classify correctly all examples in the training data, there is no guarantee that it will generalise well. In other words, nothing ensures that on a separate test set the perceptron will still perform well. The perceptron will just stop updating whenever it has found one separating hyperplane. But if the data is linearly separable, then there exists an infinity (if the data is strictly separable) of such separating hyperplanes. But they are not all as good, some have good generalisation performance, and other will have poor generalisation performance. The generalisation performance of such linear models are strongly linked with the margin of the corresponding hyperplane, and in the case of the perceptron, nothing tells us that the margin will be large enough to generalise well.

2 A General Note on Linear Models

We pause here to give general remarks on linear models. The setting is the following, we still consider a set of n observations consisting of vectors $(x_1, \dots, x_n)$ in $\mathbb{R}^d$, along with targets $(y_1, \dots, y_n)$ in $\mathbb{R}$. Here, we do not restrict the target values to binary classification only, those can be continuous values in $\mathbb{R}$ (in the case of regression) or inside a set of K possible classes, typically $1, \dots, K$ (in the case of multi-class classification). We will consider a very general loss function l that has the following properties :

- l is non-negative
- $l(y, y) = 0$ for all $y \in \mathbb{R}$
- $l(y, y') > 0$ if $y \neq y'$

We consider prediction functions of the form $x \longmapsto w^T x$ for some $w \in \mathbb{R}^d$, and try to minimize over w the empirical loss :

$$\frac{1}{n} \sum_{i=1}^{n} l(w^T x_i, y_i)$$

The first thing to note is that, if X is the design data matrix whose i-th row is x_i^T, independently of the actual expression of the loss, any w that verifies:

$$Xw = y \tag{8}$$

will minimize the empirical loss, and given the assumption on the loss it is also a necessary condition. The question now is how good such a value of w will be in terms of generalisation.

First, in order to have at least one solution to the problem, we will assume in all that follows that:

$$y \in \text{Im}(X) \tag{9}$$

Note that this assumption is implicitly realised whenever $d \geq n$ and X is full rank, *i.e.* $\text{Im}(X) = \mathbb{R}^n$. Now that we have assumed that there is at least one solution, let us look at how many solutions we can find in different cases. Having $\text{Ker}(X) = \{0\}$, *i.e.* X is injective, is actually a necessary and sufficient condition for the Equation (8) to have a unique solution under our assumption (9). Let us note that as soon as $d > n$, $\dim(\text{Ker}(X)) \geq 1$ and the linear system has an infinite number of solutions.

Now that we have set the context, let us stick with assumption (9) and not make any further assumption on n or d for now. Let w^* denote one solution to the problem. Then the space $\mathcal{S}$ of solutions to the problem is $\mathcal{S} = w^* + \text{Ker}(X)$. Let us assume $\dim(\text{Ker}(X)) \geq 1$, so that there is an infinite number of solutions. What distinguishes a good from a bad solution, *i.e.* one that will generalise well vs. one that will generalise poorly ? Let us look at the solution found by Stochastic Gradient Descent (SGD), which is the workhorse algorithm and is known to provide solutions that generalise well in the case of deep learning (massively over-parameterized models). Starting from $w_0 = 0$, SGD computes the next value of the weights as $w_{t+1} = w_t - \eta_t \frac{\partial l}{\partial u_1}(w_t^T x_{i_t}, y_{i_t}).x_{i_t}$. This shows that the solution obtained by SGD, (and actually even by GD since the update will be the sum over samples of the SGD update) is always in the linear span of the data points. So even when the linear system has an infinite number of solutions, SGD (or GD for that matter) finds one particular solution which is in the sub-space $\mathcal{S} \cap V$ where $V = \text{Span}((x_i)_{i=1,\dots n})$. We will see in what follows that this solution is actually a very particular one, and that it is in fact a solution of minimal ℓ_2 norm.

Let us forget SGD for now and go back to our original problem. However, considering what we learnt in the previous paragraph about the solution found be SGD, let us try to describe the subs-space $\mathcal{S} \cap V$ of solutions that are linear combinations of the training data points. Let $w = \sum_i \alpha_i x_i$ be such a linear combination of the data points which solves the linear system. Then $w = X^T \alpha$, and $Xw = y$, which entails that

$$XX^T \alpha = y \tag{10}$$

Conversely, any solution α to $XX^T \alpha = y$ yields a solution $w = X^T \alpha$ to Equation (8). But is there any solution to Equation (10) ? What we show below is that with the assumption $y \in \text{Im}(X)$, there is always at least one solution of Equation

(8) which is a linear combination of the data points. Indeed, let us decompose the space $\mathbb{R}^d$ as:

$$\mathbb{R}^d = \text{Ker}(X) \overset{\perp}{\bigoplus} \text{Ker}(X)^{\perp} = \text{Ker}(X) \overset{\perp}{\bigoplus} \underbrace{\text{Im}(X^T)}_{V} \tag{11}$$

and let us decompose w^* as:

$$w^* = \underbrace{w_K}_{\in \text{Ker}(X)} + \underbrace{w_I}_{\in \text{Im}(X^T)}$$

We then have that $w_I = w^* - w_K \in \mathcal{S}$, but also that $w_I = X^T \alpha_I$ for some $\alpha_I \in \mathbb{R}^n$, which implies that $w_I \in V$, and thus w_I is a solution to Equation (8) that is a linear combination of the data points (a solution to Equation (10)). It therefore follows that $Xw = y$ has a solution if and only if there is a linear combination of the data points which is a solution. What is more, we will show right now that there is a unique solution of Equation (8) in V. Indeed, if there were two solutions w_1 and w_2 both in V, then we would have $w_1 - w_2 \in Ker(X) \cap V = 0$ by (11), so that $w_1 = w_2$.

We thus see that the linear span V of the data points has a particular role when it comes to finding the solution of Equation (8). Indeed, if there is a solution, then there has to be one in V, and even when the solution is unique ($d \leq n$ and $\text{Ker}(X) = 0$), it is the solution found in V. When there are more than one solution, than there are an infinity of solutions, and $\mathcal{S}$ can be describe as $\mathcal{S} = w_V + \text{Ker}(X)$, where w_V is the unique solution in V. But since $V = \text{Im}(X^T) \perp \text{Ker}(X)$, if we decompose $w \in \mathcal{S}$ as $w = w_V + w_K$, we have that $||w||_2^2 = ||w_V||_2^2 + ||w_K||_2^2$ so that $||w_V||_2 = \min_{w \in \mathcal{S}} ||w||_2$. So, when there is a solution to Equation (8), than the solution in V is actually the solution of minimal ℓ_2 norm, which brings some regularisation to this specific solution, and makes it generalise better. Coming back to the SGD, we now see that there is actually **implicit regularization** when finding a solution to (8) with SGD, which makes this solution better than others. This seems to extend to deeper than one-layer networks, we tend to observe empirically some implicit regularisation in massively over-parameterised neural networks, which makes them able to generalise well even if the number of parameters is huge, which goes against the classical point of view of generalisation in traditional Statistical Learning Theory (VC-dimension, Rademacher complexity,etc). Additionally, we see that when the solution is unique, it is always the one in V which appears to be one that generalises better.

3 Logistic Regression

The estimator we consider here is a linear estimator of the form

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

where w and b are the learnable parameters we try to optimize.

3.1 Supervised Learning setting

The setting is that of supervised learning where we observe a finite number of data points $(x_1, x_2, ..., x_n) \in \mathbb{R}^{n \times d}$ along with the target values $(y_1, ..., y_n) \in \mathbb{R}^n$. We consider the values x_i as independent realizations of the same random variable X and the values y_i as independent realizations of the same random variable Y.

The logistic regression is the classical example of training a supervised learning algorithm : it trains to minimize the empirical risk on the data using the loss $l(y, f(x)) = \ln(1 + e^{-yf(x)})$.

In theory, the problem logistic regression tries to solve is :

$$\min_{w,b} \ \mathbb{E}[\ln(1 + e^{-Yf_{w,b}(X)})]$$

Of course, the loss quantity in the objective is not accessible as such and we can approximate it, using finite data, by :

$$\min_{w,b} \ \frac{1}{n}\sum_{i=1}^{n}[\ln(1 + e^{-y_i f_{w,b}(x_i)})]$$

The objective function is convex in the parameters w and b, and the algorithm used to solve this problem is akin to gradient descent. The Newton-Rapson method is used in practice to find an optimal couple (up to some small optimisation error in practice) (w^*, b^*).

3.2 Probabilistic view

Here we make a hypothesis on the way the data was generated : the underlying model of the logistic regression is parameterized by a vector $\theta = (w, b) \in \mathbb{R}^{d+1}$.

The model is the following :

$$\mathbb{P}(Y = 1|X = x;\ \theta) = \sigma(w^T x + b)$$

where $\sigma(z) = 1/(1 + \exp(-z))$ is the logistic function. Given this model, and since $\sigma(-z) = 1 - \sigma(z)$,

$$\mathbb{P}(Y = -1|X = x;\ \theta) = \sigma(-(w^T x + b))$$

and thus, for $y \in \{-1, 1\}$:

$$\mathbb{P}(Y = y|X = x;\ \theta) = \sigma(y(w^T x + b))$$

The idea behind this model is that we want to find a hyperplane such that when a data point x is far above it, *i.e.* $w^T x + b > 0$ then the probability of having $y = 1$ is high, and similarly when x is far below the hyperplane, *i.e.* $w^T x + b < 0$ the probability of having $y = -1$ is high. Leaving only the space close to the hyperplane where probabilities are lower. The choice of the sigmoid for the model of conditional probability is therefore a good one because $\sigma(z)$ is quickly close to 1 when $z > 0$ and quickly close to 0 when $z < 0$.

Actually, the model is such that the log of the ratio of the conditional probabilities (given x) is an affine function of x :

$$\ln\left(\frac{\mathbb{P}[Y=1|X=x]}{\mathbb{P}[Y=-1|X=x]}\right) = w^T x + b$$

which means that whenever the affine function of x is slightly positive, the ratio of the conditional probabilities is huge (thus giving way more chances to y being 1), and similarly as soon as the affine function is slightly negative then the ratio must be very small (thus giving way more chances to y being -1).

Given those assumptions, as often in the probabilistic setting, given the data, we try to estimate w and b by maximizing the log-likelihood $l_n(\theta)$:

$$\max_{\theta} \ \ln(p(y_1, \ldots, y_n | x_1, \ldots, x_n;\, \theta))$$

With the assumption that the data points are independent, we have :

$$\begin{aligned} l_n(\theta) &= \sum_{i=1}^{n} \ln(p(y_i|x_i;\, \theta)) \\ &= \sum_{i=1}^{n} \ln(\sigma(y_i(w^T x + b))) \\ &= -\sum_{i=1}^{n} \ln(1 + \exp(-y_i(w^T x + b)) \end{aligned}$$

The maximization of the log-likelihood is thus equivalent of the minimization of the the quantity above. And the **logistic regression problem thus sums up to** :

$$\min_{w,b} \sum_{i=1}^{n} \ln(1 + \exp(-y_i(w^T x + b))$$

We find that the probabilistic view solves the same problem as the original formulation of the logistic regression problem. We can observe on the formula above that for any hyperplane defined by (w, b), any point that is far form the hyperplane (e.g. $w^T x + b > 1$) and that is well classified (*i.e.* $y_i(w^T x_i + b) > 0$) does not play a big part in the objective since the corresponding loss term is very low. This shows that logistic regression is not very sensitive to outliers.

Let us conclude by noting that, exactly as in Section 7.2.1, if we add to the model a Gaussian prior on the full weight vector w, we obtain a regularized version of the logistic regression with an ℓ_2 norm penalty on w. This regularised version of the logistic regression is the one that is always (it is, for instance, the one that is implemented in scikit-learn by default) used in practice (in order to avoid over-fitting and have a lower "generalisation" error on the test set).

3.3 The Newton-Rapson algorithm for logistic regression

We describe here the method used to find (w^*, b^*) which minimize (to a small error margin) the quantity above. The first natural thing to try, since the function is convex, is to set to zero the gradient. Doing this with respect to w yields :

$$\sum_{i=1}^{n} \frac{y_i}{1 + \exp(-w^T x_i + b)} x_i = 0$$

which is intractable. The calculation with respect to b is not tractable either, and to solve $\nabla_\theta(-l_n(\theta)) = 0$ we have to use an iterative method called the Newton-Rapson algorithm, or the Iterative Re-weighted Least Squares (IRLS).

This method works by iteratively solving least square problems. We start from a random θ_0 and, at each step t minimize the second order (convex) Taylor expansion of $-l_n(\theta)$ by solving a least square problem. If we denote by $\mathcal{H}(\theta)$ the Hessian matrix of $-l_n(\theta)$ at point θ, we find, at each step, θ_{t+1} by minimizing with respect to θ the quantity :

$$-l_n(\theta_t) - \nabla_\theta(l_n)(\theta_t)(\theta - \theta_t) - (\theta - \theta_t)^T \mathcal{H}(\theta_t)(\theta - \theta_t)$$

As we will see with the least square regression, this minimization problem does not always have a unique solution (if $\mathcal{H}(\theta_t)$ is not invertible), but **regularizing** the logistic regression by adding an ℓ_2 norm penalty to the objective yields an additional term in the Hessian proportional to the identity matrix and makes the least square problem solvable with a unique solution.

3.4 Regression with logistic regression

Although the probabilistic model underlying the logistic regression only makes sense in the setting of classification problems, the algo itself and its formulation are perfectly valid for regression : estimating a continuous variable $y \in \mathbb{R}$ from $x \in \mathbb{R}^d$.

However, in contrast, the probabilistic model makes much more sense for the classification problem and justifies using the sign of the regressor to classify examples.

4 Linear Regression with Least Squares

In this section we consider once more linear classifiers of the form $f_{w,b}(x) = w^T x + b$. We consider a data matrix $X = [x_1; x_2; ...; x_n] \in \mathbb{R}^{n \times d}$ along with the target values $y = (y_1, ..., y_n) \in \mathbb{R}^n$.

4.1 The supervised learning setting

Here we consider the mean squared error and try to find the parameters (w, b) which minimize the average squared error between the linear prediction and the target :

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} (y_i - (w^T x_i + b))^2$$

To simplify notations, we add the offset to the vector parameterization and consider the vector $\beta = (w, b)$ and optimize over β. Adding a fictitious last dimension to the vectors of inputs x, whose value is always 1, we are reduced to solving :

$$\min_{\beta} \left(\frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 = \frac{1}{n} ||y - X\beta||_2^2 \right)$$

or, equivalently

$$\min_{\beta} \; ||y - X\beta||_2^2$$

which is known as a **least squares problem**. This is a convex problem and the solution is found by looking for values which make the differential null. The solution is found by solving the system $X^T X \beta = X^T y$, which can be computed in closed form if $X^T X$ is invertible. If it is not, than more than one solution might exist but an inverse (less restrictive than what a real inverse would be) can still be computed and solutions obtained from it. The matrix $X^T X = \sum_i x_i x_i^T$ is not invertible as soon as $d > n$, or typically when the features are highly correlated since X will not be of full rank.

Indeed, if $d > n$, then $u \longmapsto X^T X u$ is a linear application from $\mathbb{R}^d$ to $\mathbb{R}^d$, and is the composition of the two linear applications $v \longmapsto X^T v$, from $\mathbb{R}^n$ to $\mathbb{R}^d$, and $u \longmapsto Xu$, from $\mathbb{R}^d$ to $\mathbb{R}^n$. However, for any two linear applications g and h, $\text{rank}(h \circ g) \leq \text{rank}(g)$ since for any space F, $\dim(h(F)) \leq \dim(F)$. We thus have that $\text{rank}(X^T X) \leq \text{rank}(X) \leq n < d$, which implies that $X^T X$ is not of full rank and therefore non-invertible.

Now let us assume that $n \geq d$, and that the features are correlated, which is to say that there exists $(\lambda_1, \dots, \lambda_d)$ such that for any data point $x \in \mathbb{R}^d$, $\sum_{k=1}^{d} \lambda_k \, x(k) = 0$ where $x(k)$ denotes the k-th entry of the vector x. Then the d columns of X are linearly dependent. However, the rank of X is the dimension of the linear of its columns, and the latter space is of dimension less than $d-1$. Then, with the same argument as in the previous paragraph we prove that $X^T X$ is not invertible.

What we show here is that even when the matrix $X^T X$ is not invertible, we can still find at least one solution to the problem, and we even show that there are in fact an infinity of solutions (the problem is under-determined). Indeed, $X^T X$ is a positive semi definite matrix, and if it is not invertible, it will have at least one eigenvalue which is 0. More than that, let us decompose the euclidean space $\mathbb{R}^d$ as :

$$\mathbb{R}^d = V \overset{\perp}{\bigoplus} V^{\perp}$$

where $V = \mathrm{Span}\big(x_1, \dots, x_n\big)$. Then, it is easy to see that we actually have $V^{\perp} = \mathrm{Ker}(X^T X)$, since, by writing $X^T X = \sum_i x_i x_i^T$, for $u \neq 0$,

$$X^T X u = 0 \Longleftrightarrow u^T X^T X u = 0 \Longleftrightarrow \sum_i (x_i^T u)^2 = 0 \Longleftrightarrow \forall i, u \perp x_i$$

Therefore, since $X^T X$ is symmetric we also have that $V = \mathrm{Im}(X^T X)$. What is more, since $X^T y = \sum_i y_i x_i \in V$, we see that there is always a solution to the least square problem, even when $X^T X$ is not ivertible. To find a solution to the problem, we can compute the inverse of the invertible part of $X^T X$. Indeed, when diagonalising $X^T X$ in an orthonormal basis, we have $X^T X = ODO^{-1}$ where O is an orthogonal matrix and D is a diagonal matrix with some positive values on the diagonal and at least one 0 on the diagonal when $X^T X$ is not invertible. By inverting the eigenvalues which are positive on the diagonal we find the pseudo inverse $(X^T X)^{\dagger}$ of $X^T X$. We get one solution of the problem by computing $(X^T X)^{\dagger} X^T y$. And then, if $X^T X$ is not invertible, this solution is not unique since we can add to it any vector in $V^{\perp}$ to get another solution, such that there are, in fact, an infinite number of solutions when $X^T X$ is not invertible.

The problem of overfitting occurs as soon as the minimal (positive) eigenvalue of $X^T X$ is small, since it will cause great instability in solving the system, $\beta = (X^T X)^{\dagger} X^T y$. In fact, the maximal eigenvalue of $(X^T X)^{\dagger}$ can be very large if the minimal eigenvalue of $X^T X$ is too small, and a small change in $X^T y$, *i.e.* in the training data, can lead to a totally different weight vector β. As we will see in Section 5, regularization can help reduce overfitting because it ensures that the minimal eigenvalue of $X^T X$ is some $\lambda > 0$, thus allowing for more stability for the solution weights with respect to the training data.

4.2 The general least squares problem

We make a small digression here on the form of the optimal for a least squares (not linear, the general problem) regression problem. Let us consider the random variables X (which thus does not denote the data matrix in this paragraph) and Y in the least squares regression problem. The generalization error associated with the least squares regression is $\mathbb{E}[(Y - f(X))^2]$, which we try to minimize over all possible choices of f's. This error writes as :

$$\mathbb{E}[(Y - f(X))^2] = \mathbb{E}\big[\mathbb{E}[(Y - f(X))^2 \,|X]\big]$$

and for any given x,

$$\operatorname*{argmin}_{a \in \mathbb{R}} \mathbb{E}[(Y - a)^2 \,|X = x] = a^* = \mathbb{E}[Y|X = x]$$

which yields that the optimal f for this problem is :

$$f^*(x) = \mathbb{E}[Y|X = x]$$

Would we be able to compute this quantity for any x we would have solved the problem. However, we do not have directly access to this quantity in practice, and we can try to estimate it. For instance, the k-nearest neighbours algorithm tries to estimate this quantity directly, and should be able to get close to the optimal predictor. However, it is essential to note that in very-high dimensional spaces (tens of thousands of dimensions), the "closet" neighbours of a given data point x_i are usually very far from it, so that the estimation of $\mathbb{E}[Y|X = x_i]$ by the average value of y for the k nearest neighbours of x_i is very far from being accurate, and the number of data points needed to make it an acceptable estimation is huge, which makes this type of local estimation methods flawed in very high dimension. Thus the problem with high dimension is that data points are all far from each other, and the space is actually "very empty".

4.3 The probabilistic point of view

Here we consider a different point of view where we model the way the data was generated : we consider that, given $X = x$, Y is drawn from a Gaussian distribution with mean $\beta^T x$ and variance σ^2. In other words, we have:

$$Y = f(X) + \varepsilon$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. In this setting we thus consider a conditional probabilistic model, parameterized by $\theta = (\beta, \sigma^2)$. We have that :

$$p(y|x;\, \theta) = \mathbb{P}[Y = y|X = x;\, \theta] = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\big(-\frac{(y - \beta^T x)^2}{2\sigma^2}\big)$$

and we estimate the parameters β and σ^2 using the maximum likelihood principle. We seek to maximize the log of the conditional probability of the observations :

$$\max_{\beta,\sigma^2} \log(p(y_1,\dots,y_n|x_1,\dots x_n;\theta))$$

which, after calculation, boils down to solving:

$$\max_{\beta,\sigma^2} -\frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta^T x_i)^2$$

and thus equivalently :

$$\min_{\beta,\sigma^2} \frac{n}{2}\log(\sigma^2) + \frac{1}{2\sigma^2}||y - X\beta||_2^2$$

which looks very much like the linear least square regression formulation in the pure supervised learning setting, except that now a variance coefficient σ^2 appears, due to the addition of this parameter in the model. However, the minimization in β still yields the same solution(s) characterized by the same equation. Optimizing over σ^2 we find $\widehat{\sigma^2} = \frac{1}{n}||y - X\hat{\beta}||_2^2$.

5 Ridge Regression

The Ridge regression is a regularized version of the linear regression with least squares where we add a penalty equal to the squared euclidean norm of the weight vector (up to a constant). The Ridge regression problem is thus, for a fixed $\lambda > 0$:

$$\min_{w,b}\sum_{i=1}^{n}(y_i - w^T x_i - b)^2 + \lambda||w||_2^2$$

Trying to shrink the norm of the learned weight vector allows to avoid “crazy” correlations to specific data : the coefficients of the vector w could take very large values to minimize the loss by adapting a lot to the data at hand. Forcing those coefficients not to be to large reduces that effect. More formally, the euclidean norm of the weight vector is directly related to the variance of the corresponding estimator, and the problem we want to solve is equivalent to :

$$\min_{w,b}\frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i - b)^2$$

$$\text{s.t. } ||w||_2^2 \leq t$$

for a certain t, which is directly related to the constant λ through the data. Therefore, the problem in itself has not changed, and the objective to minimize is still the same : find the best linear predictor to fit the data in terms of squared difference, but this time, we restrict the search for a w to a certain ball centered at 0. Let us note that if we consider the original formulation, the invertibility issue with X^TX in the non-regularized form is automatically bypassed since, now, the solution in w yields :

$$2(X^TX + \lambda I_d)w^* = 2(y - b^*)$$

Thus, adding regularization ensures an invertible system to solve (since the matrix $X^TX + \lambda I_d$ is positive definite, it is invertible), leading to :

$$w^* = (X^TX + \lambda I_d)^{-1}(y - b^*)$$

Given that the eigenvalues of the matrix $(X^TX + \lambda I_d)^{-1}$ are smaller than those of X^TX, the learned weight vector for regularized least squares will have smaller coefficients than the one learned via usual least squares. (this is even more obvious when considering the SVD decomposition of X^TX -if not invertible- or when in the orthogonal case where $x_i^T x_j = \delta_{i,j}$, where δ is Kronecker's symbol).

Let us simplify the analysis of the variance of the learned weight vector by assuming the data matrix X to be given, and the random target variable Y to have variance σ^2. The covariance matrix of w^* under those assumptions is thus:

$$\text{Var}[w^*] = \sigma^2(X^TX + \lambda I_d)^{-1}$$

which has again smaller eigenvalues than that of what the covariance matrix would be (namely $\sigma^2(X^TX)^{-1}$) under the same assumptions in the non-regularized case.

6 Lasso and ℓ_1 regularization

The LASSO algorithm (Least Absolute Shrinkage and Selection Operator) algorithm is another least squares regression problem where this time the penalization used a ℓ_1 norm. Meaning that the problem we wish to solve is, for a fixed $\lambda > 0$:

$$\min_{w,b} \sum_{i=1}^{n} (y_i - (w^Tx_i + b))^2 + \lambda||w||_1$$

or, equivalently, finding the minimal value within a ball in norm ℓ_1, for some $t > 0$, which is related to λ via the data :

$$\min_{w,b} \sum_{i=1}^{n} (y_i - (w^Tx_i + b))^2$$

$$\text{s.t. } ||w||_1 \leq t$$

The effect of the LASSO is twofold :

1. the learned estimator will have a lower variance than that of Ridge Regression (i.e. ℓ_2 penalized least squares), and thus better generalization capabilities
2. the ℓ_1 penalization will tend to set some of the coefficients of the weight vector to zero, thus allowing for higher interpretability in the model (which dimensions in the input bear the most weight in the prediction decision)

We give below informal intuition as to why those two main properties hold. We will start with the second property. To better form our intuition we will consider the case in dimension $d = 2$. The level sets of the function

$$: \beta = (\beta_1, \beta_2) \longmapsto \sum_{i=1}^{n} (y_i - \beta^T x_i)^2$$

are concentric ellipses, centered around the minimal attainable value $\widehat{\beta}^{\mathrm{OLS}}$ which is the value found when solving the ordinary least squares (OLS) problem (*i.e.* without any regularization). The further these ellipses are from the center, the higher the corresponding value of the function is. For a given value of t, the ℓ_1 and ℓ_2 penalized least squares linear regression write as :

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2$$
$$\text{s.t. } ||\beta||_1 \leq t$$

and

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2$$
$$\text{s.t. } ||\beta||_2^2 \leq t$$

The constraints $||\beta||_1 \leq t$ and $||\beta||_2^2 \leq t$ imply that we look for a value inside a ball centered at 0, in norm ℓ_1 and ℓ_2 respectively. We can represent, for a given t (e.g. $t = 1$) theses balls in 2d. The goal is to find the closest ellipse to the center which has a point inside one of the two balls. The only way for an ellipse to be tangent to the ball in norm ℓ_2 at one of the 4 $(0, 1), (1, 0), (-1, 0), (0, -1)$ (which are the points of the sphere $||x||_2^2 = 1$ with one null coordinate) is to be an axis-orthogonal ellipse. In contrast, ellipses with many different inclinations can hit the sphere $||x||_1 = 1$ in exactly one of the 4 points above. The Figure 1 below makes this informal argument clearer.

More formal proof of this result can be obtained by writing the form that the optimal solution β^* of the ℓ_1-regularized (penalized) linear regression problem which will reveal that naturally some of the coefficients will tend to be equal to 0. Yet another intuitive explanation is that, as described in Section 7.2, the ℓ_1

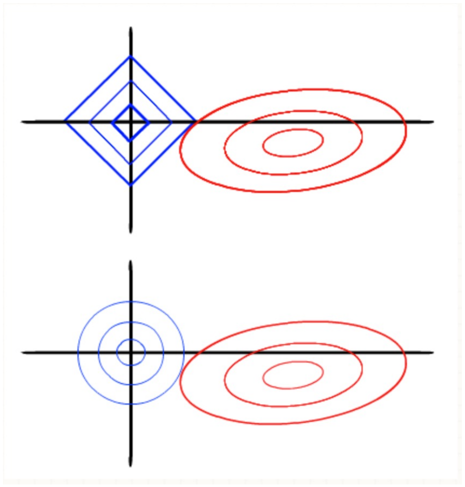

Figure 1: ℓ_1 vs ℓ_2 regularization

penalty in the Lasso can be seen as a Laplace prior on β, which is more peaked and sharp around 0 then the Gaussian distribution which an ℓ_2 penalty (as in Ridge regression) assumes. Therefore, the probability of some coefficients being exactly zero is much higher under this model.

As for the first property, it is not obvious why the ℓ_1 regularization should reduce the variance. One argument we put forward is that, as previously stated, ℓ_1 regularization shrinks some coefficients of the learned weight vector to 0 and thus acts as a subset selector from the initial coefficients, which will help reduce variance.

7 L1 vs. L2 Regularization

7.1 Differences and impact of both algorithms

Use Ridge instead of Lasso when variables are highly correlated, prefer Lasso (in general) when there are a lot of features. Also choose Lasso instead of Ridge if you want higher explainability of the model : feature selection (which variables matter the most for predicting the target ?).

7.2 Laplace vs Gaussian prior

Let us take a look at the least square problem with ℓ_1 and ℓ_2 regularization from a Bayesian probabilistic standpoint. As we will show below, the ℓ_2 regularization corresponds to putting a Gaussian prior on the weight vector β whereas the ℓ_1 regularization corresponds to putting a Laplace prior the weight vector.

The probabilistic setting with prior is the same as the one presented in Section 4.3 where we consider that the random variables X and Y are related by the following equation :

$$Y = \beta^T X + \varepsilon$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, so that given $X = x$, $Y|X = x \sim \mathcal{N}(\beta^T x, \sigma^2)$. This formulation is equivalent to the classic least squares formulation and yields the same solution(s) for the weight vector β, but the probabilistic version adds one more parameter to be estimated : the variance parameter σ^2 of the Gaussian law.

7.2.1 Gaussian prior and ℓ_2 regularization

Now let us add to the original unregularized least square probabilistic problem by putting a Gaussian prior on β. More precisely, we assume that β follows a normal law $\mathcal{N}(0, \alpha^2 I_d)$ where α is a fixed parameter, and is independent of ε and X. Now, given $\beta = \bar{\beta}$ and $X = x$, $Y \sim \mathcal{N}(\bar{\beta}^T x, \sigma^2)$. Given a dataset of observations $x = (x_1, \dots, x_n)$ which we represent by the usual data matrix $\mathbf{X} = [x_1^T; \dots; x_n^T] \in \mathbb{R}^{n \times d}$ and $y = (y_1, \dots, y_n) \in \mathbb{R}^d$, we can write the full conditional log-likelihood on y and β :

$$\begin{aligned}
l_n(\beta, \sigma^2) &= \sum_{i=1}^{n} \log p(y_i, \beta | x_i; \sigma^2, \alpha^2) \\
&= \sum_{i=1}^{n} \log p(\beta|\alpha^2) + \log p(y_i | x_i, \beta; \sigma^2) \\
&= -\frac{n}{2\alpha^2}||\beta||_2^2 + \frac{n}{2}\log(1/\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta^T x_i)^2 + c^{te} \\
&= -\frac{1}{2}\Big[\frac{1}{\sigma^2}||\mathbf{X}\beta - y||_2^2 + \frac{n}{\alpha^2}||\beta||_2^2 - n\log(1/\sigma^2)\Big] + c^{te}
\end{aligned}$$

and thus, the maximization of the complete conditional log-likelihood with respect to β and σ^2 amounts to solving the following minimization problem :

$$\min_{\beta, \sigma^2} \frac{1}{\sigma^2}||\mathbf{X}\beta - y||_2^2 + \frac{n}{\alpha^2}||\beta||_2^2 - n\log(1/\sigma^2)$$

which is nothing else than the probabilistic version of the ℓ_2 regularized least square problem. For a given $\lambda > 0$, taking $\alpha = \frac{\sqrt{n}\sigma}{\sqrt{\lambda}}$, the objective to minimize becomes :

$$\frac{1}{\sigma^2}\Big[||\mathbf{X}\beta - y||_2^2 + \lambda||\beta||_2^2\Big] - n\log(1/\sigma^2)$$

and, since the solution in β is decoupled from the solution in σ^2, we get that β is the solution of the original ℓ_2 regularized least squared problems with penalty λ.

7.2.2 Laplace prior and ℓ_1 regularization

Let us look at a similar line of arguments but this time placing a Laplace prior on β with mean parameter 0 and scaling parameter α. Let us recall the expression of the Laplace distribution with mean parameter μ and scaling parameter $\alpha > 0$, the density of the law at point x is :

$$L(x;\mu,\alpha) = \frac{1}{2\alpha}\exp\left(- \frac{||x-\mu||_1}{\alpha}\right)$$

We can still decompose the conditional log-likelihood as before, and changing the form of the prior changes only one term, yielding, for a fixed α :

$$l_n(\beta,\sigma^2) = -\frac{n}{\alpha}||\beta||_1 - \frac{1}{2\sigma^2}||\mathbf{X}\beta - y||_2^2 + \frac{n}{2}\log(1/\sigma^2) + c^{te}$$

The maximization of the log-likelihood with respect to β and σ^2 is thus equivalent to the minimization problem :

$$\min_{\beta,\sigma^2} \frac{1}{\sigma^2}||\mathbf{X}\beta - y||_2^2 + \frac{2n}{\alpha}||\beta||_1 - n\log(1/\sigma^2)$$

For a fixed value of $\lambda > 0$, taking $\alpha = \frac{2n\sigma^2}{\lambda}$, we end up solving the following problem :

$$\min_{\beta,\sigma^2} \frac{1}{\sigma^2}\Big[||\mathbf{X}\beta - y||_2^2 + \lambda||\beta||_1\Big] - n\log(1/\sigma^2)$$

and again, the solution in β is then no other than the solution to the classic ℓ_1 regularized least squares problem. We note that the Laplace distribution is sharper than the Gaussian distribution around 0 and will thus tend to give higher probability to points which have some coefficients equal to 0 compared to the Gaussian distribution, which explains why the LASSO has sparse solutions.

8 SVM

We consider again here linear classifiers $f_{w,b}(x) = w^Tx+b$, but this time there is no probabilistic model underlying the choice of loss. Instead, we see things from a more geometric perspective, and try to find a separating hyperplane for which the minimum distance from points in the dataset is as big as possible (that is the idea in the separable case, at least), hoping that by doing so the generalization capabilities of the learned classifier will be good (since the distance from the hyperplane to its closest point is large, we hope that new, unseen, data will still fall on the right side of the hyperplane because the margin is big, which is to say that the separating hyperplane is neat and far from the data points on both sides, such that even if the data changes a little they still get classified correctly).

8.1 The seperable case

Let us now formalize this idea : given a weight parameter $w \in \mathbb{R}^d$ and and intercept $b \in \mathbb{R}$, we define the hyperplane $\mathcal{H}_{w,b}$ by the equation $w^T x + b = 0$. The distance between any point $x \in \mathbb{R}^d$ and $\mathcal{H}_{w,b}$ is :

$$d(x, \mathcal{H}_{w,b}) = \frac{|w^T x + b|}{||w||_2}$$

And any point x is classified using $\text{sgn}(w^T x + b)$. The goal is thus to find a separating hyperplane, *i.e.* one for which $\forall i \in \{1, \dots, n\},\ y_i(w^T x_i + b) > 0$, which at the same ensures to find w such that the minimal distance to the hyperplane $\min_i d(x_i, \mathcal{H}_{w,b})$ is maximal :

$$\max_{w,b} \min_{i=1,\dots,n} \frac{|w^T x_i + b|}{||w||_2}$$

$$\text{under the constraint that } \forall i \in \{1, \dots, n\},\ y_i(w^T x_i + b) > 0$$

Given the constraints, we can rewrite the distance without the absolute values: $|w^T x_i + b| = y_i(w^T x_i + b)$. Which yields :

$$\max_{w,b} \frac{1}{||w||_2} \min_i y_i(w^T x_i + b)$$

$$\text{s.t. } \forall i,\ y_i(w^T x_i + b) > 0$$

8.1.1 Solving the primal

The optimization problem as such is not solvable using off-the-shelf optimization solvers, and we have thus to reformulate it to obtain a form which allows to use directly optimization solvers. The choices we will make in order to rewrite this problem come from different attempts to rewrite it differently (which we will not detail, but are very well explained in Stanford's CS229 course by Andrew Ng[1]). Let us however make two observations to explain where problems lie : first the denominator $||w||_2$ in the distance makes the problem non-convex and thus difficult to solve. We could remove this issue by enforcing $||w||_2 = 1$, or equivalently $||w||_2^2 = 1$ but then this term would appear in the constraints and it is not a linear constraint, which again makes the problem more difficult to solve.

What we can note however is that the problem does not change up to some reparameterization : for instance, we can multiply b or w by any constant without changing the objective of the problem at hand. What we will thus do is force a parameterization of the problem where the quantity $\min_i y_i(w^T x_i + b)$,

[1]http://cs229.stanford.edu/notes/cs229-notes3.pdf

called the functional margin, is equal to 1. It is no completely obvious that the new problem :

$$\max_{w,b} \frac{1}{||w||_2}$$

$$\text{s.t. } \forall i,\ y_i(w^T x_i + b) > 0,\ \text{ and } \min_i y_i(w^T x_i + b) = 1$$

is formally equivalent to the old one, but it takes a few calculations to convince oneself (the new problem is more restrictive than the old, but any solution to the old problem can be transformed into a solution which satisfies the constraints of the new problem). We can one last time rewrite the constraints to make the min disappear and get a nicer formulation :

$$\max_{w,b} \frac{1}{||w||_2}$$

$$\text{s.t. } \forall i,\ y_i(w^T x_i + b) > 0,\ \text{ and } y_i(w^T x_i + b) \geq 1$$

which, in turn leads to the final formulation :

$$\min_{w,b} \frac{1}{2}||w||_2^2$$

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

which is a convex problem with linear constraints known as a **Quadratic Program** for which we have efficient solvers. What we learn from this formulation is that at the end of the day, the problem we are trying to solve is to find the w with smallest norm such that the prediction on any x_i from the dataset is greater, in absolute value, than 1, and of the same sign as y_i. In other words we find again that the optimal classifier should give a high margin and thus a good confidence in the predictions on the dataset. Unfortunately, this kind of solvers become way less efficient as the dimension d of the data grows large. We will now introduce the dual formulation of the problem and see what insights and advantages can be drawn from it.

8.1.2 The dual formulation

Before writing the dual, let us first write the Lagrangian :

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}||w||_2^2 + \sum_{i=1}^{n} \alpha_i(1 - y_i(w^T x_i + b))$$

The dual writes as :

$$\max_{\alpha \geq 0} \left(g(\alpha) = \min_{w,b} \mathcal{L}(w,b,\alpha)\right)$$

whereas the primal writes as :

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

The interesting point to note here is that since the objective is convex, the inequality constraints are convex and strictly feasible (there are no equality constraints, but if there were we would require them to be linear to satisfy the conditions of the theorem), we have that :

1. both the primal and the dual have optimal solutions which we denote respectively by (w^*, b^*) and α^* (which thus satisfy the constraints)
2. the primal and the dual values are equal, and both equal to $\mathcal{L}(w^*, b^*, \alpha^*)$
3. in addition, the optimal solutions w^*, b^*, α^* satisfy the **Karush, Kuhn and Tucker (KKT) conditions** :

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_l}(w^*, b^*, \alpha^*) &= 0 \\
\frac{\partial \mathcal{L}}{\partial b}(w^*, b^*, \alpha^*) &= 0 \\
\alpha_i^*(1 - y_i(x_i^T w^* + b^*)) &= 0 \quad \text{(complementary slackness)}
\end{aligned}$$

The most interesting equation is the second one, called **complementary sclackness**, which tells us that the Lagrange multiplier must be zero for any data point which does not achieve the minimal margin ($i.e.$ $y_i(x_i^T w^* + b^*) > 1$). The points which achieve **the minimal margin equality** $y_i(x_i^T w^* + b^*) = 1$ are called the **support vectors** and we will see below that the learned solution (w^*, b^*) depend only on those, which is the main result of considering the problem from the dual perspective.

Let us take a closer look at the dual. More precisely, let us look at the function $g(\alpha)$ which we wish to maximize. It is essential to observe that the minimization problem defining $g(\alpha)$ is decoupled in w and b, which means that we can minimize over both variables separately. In doing so, what we see is that if a given α does not satisfy $\sum_{i=1}^n \alpha_i y_i = 0$, then the minimization in b yields $-\infty$, so that $g(\alpha) = -\infty$ whenever $\sum_{i=1}^n \alpha_i y_i \neq 0$. Therefore, the dual maximization problem reduces to :

$$\max_{\alpha \geq 0} \; \min_{w} \frac{1}{2}||w||_2^2 - \sum_{i=1}^n \alpha_i y_i w^T x_i$$

$$\text{s.t.} \; \sum_{i=1}^n \alpha_i y_i = 0$$

The minimization involved is that of a (strictly) convex function over a convex an open set, and can be done by setting the partial derivatives to zero, which yields, for any α satisfying the constraints just above :

$$g(\alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

and thus leads to the following formulation for the dual :

$$\min_{\alpha \geq 0} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{n} \alpha_i$$

$$\text{s.t.} \sum_{i=1}^{n} \alpha_i y_i = 0$$

This is again a quadratic program (quadratic convex function to optimize under linear constraints). What have we then gained compared to the primal formulation ? Well the dimension of the problem is now n instead of d, which means that for problems for which the dimension of the data is huge, and widely greater than the number n of samples, the QP solver will be much more efficient when solving the dual than the primal. What is more, when solving the dual, we know for a fact (from complementary slackness conditions) that the number of α_is which are not zero is equal to the number of support vectors x_i such that $y_i(x_i^T w^* + b) = 1$. We will see just now that those α_is are directly related to the way the optimal weights w^* are connected to the data points x_i. From the KKT conditions mentioned above we know that the gradient of the Lagrangian with respect to the w and b should be 0 at the point (w^*, b^*, α^*). When we calculate the derivative with respect to b and set it to 0, we get that :

$$\sum_{i=1}^{n} \alpha_i^* y_i = 0$$

which is no surprise and we actually already new from what we said above. When we calculate the gradient with respect to w and set it to 0 we get :

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

which means that the optimal w^* is a linear combination of the data points $x_1, \ldots, x_n$. Even better, from what we know on the α_i^*, w^* is in the linear span of the support vectors only, and thus depends on only a few of the data points, which is another major advantage of solving the dual instead of the primal.

To conclude on the dual formulation of the SVM problem, we can notice that the objective we need to solve depends not on the data points x_i themselves, but on the inner products between them. Similarly, once we learned the values of w and be, to make a prediction given a new data point x we calculate :

$$x^T w^* + b^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i^T x + b$$

a quantity which, again, needs only a few of the data points (the support vectors) to be calculated, and needs only the inner product between any point and the support vectors to be known which can give rise to different algorithms where the space in which lies the x_is can be changed as long as we define a proper inner product on it. This is the basis for many Kernel methods, where computing a embedding in a higher dimensional space can be quite costly, but computing the inner product between two vectors of that space might be computed with much less effort.

8.2 The non-separable case

We consider the case here where the data is not separable, *i.e.* when there is no hyperplane separating the two class of labels perfectly : for any hyperplane, there would be data from both classes lying on the same side of the hyperplane. In other terms, the data from both classes would be mingled and impossible to set apart using one hyperplane. That being said, we can still look for a hyperplane for which we have a maximum number of samples classified with a high margin (and thus with high confidence), *i.e.* we want a maximum number of samples to satisfy $y_i(w^T x_i + b) \geq 1$. We know that we cannot have this satisfied for all the data points in the training set since the data is not separable. Yet, we can look for a classifier which has a maximum number of samples satisfying the condition, and, for those who do not, try to make it so that they are as close as possible to satisfying it.

More formally, let us consider a data point (x_i, y_i), and a classifier given by the parameters (w, b). If $y_i(w^T x_i + b) \geq 1$ then the example is very well classified with a high margin, and it does not matter by how much $y_i(w^T x_i + b)$ is greater than 1, what matters is that the confidence is high enough. So this example does not need to be penalized at all. Now assume that $y_i(w^T x_i + b) < 1$, so that we are below the margin of high confidence. One of two things can happen : either the example is still well classified $y_i(w^T x_i + b) > 0$, but with a confidence that is not satisfactory because it is below the acceptable margin (it is still too close to the hyperplane, and a variation in the data could lead to a misclassification), or $y_i(w^T x_i + b) \leq 0$ and the example is misclassified. In any case, what we want to do is bring the value of $y_i(w^T x_i + b)$ as close to 1 as possible to ensure a good margin (and not for that particular data point, but for each of those). To do so, we penalize this sample by a using the cost $1 - y_i(w^T x_i + b)$ whose sum over the data set we will try to minimize. In other words, considering a given data set of points and labels, and denoting $l(f(x), y) = \max(0, 1 - y_i f(x_i))$, we add a penalty for every point in the data set which is $l(f_{w,b}(x_i), y_i)$, where in this case $f_{w,b}(x) = w^T x + b$.

We ended up finding this loss after the informal discussion above. To be more precise, let us consider the perspective of optimization. Let us go back to the original primal formulation :

$$\min_{w,b} \frac{1}{2}||w||_2^2$$

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

The issue is that, in the non-separable case, there are no w, b satisfying the constraints. A classic way to get around this in optimization is to **relax the constraints** : demanding that every data point in the training set be classified with high margin is too much to ask in this case. What we can ask is that, the constraint on the margin be satisfied up to a certain penalty, which we will want to be as small as possible. In other words, we will introduce (non-negative) **slack variables** $\xi_i \geq 0$, and ask that $y_i(w^T x_i + b) \geq 1 - \xi_i$ instead of $y_i(w^T x_i + b) \geq 1$. And we want at the same time those slack variables ξ_i to be as small as possible, only to make it so that the constraints are feasible, which is why we add the term $\sum_i \xi_i$ in the objective, multiplied by a positive constant C which handles how importance to give to minimizing that sum. This results in a new problem for the non-separable case :

$$\min_{w,\, b,\, \xi}\ \frac{1}{2}||w||_2^2 + C\sum_{i=1}^n \xi_i$$

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

Note that the new dimension of the problem which we are trying to optimize is now $d + n$ (or rather $d + 1 + n$ to be completely correct). We can once more try to gain insights from the dual formulation. The Lagrangian writes as :

$$\mathcal{L}(w, b, \alpha, \beta) = \frac{1}{2}||w||_2^2 + C\sum_{i=1}^n \xi_i + \sum_{i=1}^n \alpha_i(1 - \xi_i - y_i(w^T x_i + b)) + \sum_{i=1}^n \beta_i \xi_i$$

and the dual as :

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

where, in the constraints, we mean by $\alpha, \beta \geq 0$ that both $\alpha \geq 0$ and $\beta \geq 0$. As, in the separable case, it is easy to see that if $\sum_i \alpha_i y_i \neq 0$ or $\sum_i(\alpha_i + \beta_i) \neq 0$, then the minimization of the Lagrangian yields $-\infty$. The dual can thus be reformulated as :

$$\max_{\alpha,\beta \geq 0}\ \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\text{s.t. } \sum_i \alpha_i y_i = 0, \text{ and } \forall i,\ \alpha_i + \beta_i = C$$

and after final reformulation :

$$\min_{\alpha} \ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{n}\alpha_i$$

$$\text{s.t.} \ \sum_i \alpha_i y_i = 0, \text{ and } \forall i, \ 0 \le \alpha_i \le C$$

The dual is till very similar to the one of the separable case. Interestingly, it still is a quadratic program optimization problem in **dimension n only**, when the primal optimized over $n+d+1$ variables. Again, the x_is appear in the objective only through their inner products $x_i^T x_j$. In addition, the primal is still in this case a convex (quadratic) optimization problem with linear constraints, and the dual and primal values are equal, and the KKT conditions hold. It takes only a few calculations, very similar to the separable case, to see that the optimal solution w^* depends only on the support vectors which are here defined by $y_i(x_i^T w^* + b) = 1 - \xi_i$. We have that :

$$\forall i, \ \alpha_i^*(1 - \xi_i^* - y_i(x_i^T w^* + b^*)) = 0$$

and

$$\forall i, \ \beta_i^* \xi_i^* = 0$$

and

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

Here again, w^* is in the linear span of the support vectors only, and the corresponding learned prediction function at a point x depends not on x or the x_is themselves but on their inner products, which allows for the use of Kernel methods.