# Support Vector Machines - Passerini

Mattia Carolo - @Carolino96

## Support Vector Machines

Support Vector Machine, SVM from now on, are linear classifiers that separate using a **large margin classifier** which solution depends only on a small subset of the traing examples called **support vector**. It's very important to note that it has a sound generalization theory (not to study) and they can be easily extended to non linear separation retaining the separation properties thanks to *kernel machines*.

## Maximum margin Classifier & Hard Margin SVM

Let's try to formalize the margin. We already know that $yf(x)$ is the confidence on the correct prediction, if negative the prediction is wrong otherwise if positive correct and the value is the confidence on the prediction. Now suppose we have a classifier that correctly separates with no training errors. If this is the case the minimum value among the training examples is called *confidence margin* and it's written like

$$\mathbb{R} = \min_{(\mathbf{x},y)\in D} yf(\mathbf{x})$$

Since it depends on $w$ we can compute the distance from the minimal distance to our classifier and it's called **geometric margin** which is formalized like

$$\frac{\mathbb{R}}{\|\mathbf{w}\|} = \min_{(\mathbf{x},y)\in D} \frac{yf(\mathbf{x})}{\|\mathbf{w}\|}$$

Ideally we want to maximize the last formula in order to get $w$ in order to maximize the margin. However if we put in an optimization problem we have actually one degree of freedom that is being removed. Suppose we have a solution where

$$\mathbf{w}^T\mathbf{x} + w_0 = 0$$

now if we want to characterize further the plane we can, for example, multiply the terms with an $\alpha \neq 0$ and still we will return to a formula that look like before since we can incorporate the $\alpha$ in our formalization. This is because there is an infinite number of equivalent formulation for the same hyperplane even with different parameters.

We can counter this problem through the introduction of the *canonical hyperplane* in which we set the constraint that $\mathbb{R}$ must be equal to a number given a priori (in our case we take 1) in order to get:

$$\mathbb{R} = \min_{(\mathbf{x},y)\in D} yf(\mathbf{x}) = 1$$

and it's geometric margin will be $\dfrac{\mathbb{R}}{\|\mathbf{w}\|} = \dfrac{1}{\|\mathbf{w}\|}$

the numerical value in the geometric margin must match

As we can see from the image above the two dotted lines are the two canonical hyperplanes with their $\mathbb{R}$ set to 1 so summing their respective geometric margin we get that the total geometric margin is equal to $\dfrac{2}{||\mathbf{w}||}$.

We can take this and convert it to an optimization problem.

First of all we want to maximize the margin so $\dfrac{2}{||\mathbf{w}||}$ and we want to do it by enforcing all examples to stay on the correct part of the hyperplane for both canonical ones. Formalized will be

$$\max \frac{2}{||\mathbf{w}||} \text{s.t.} \forall x_i : y_i = 1 \Rightarrow \mathbf{w}^T x_i + w_0 \geq 1 \;\; \& \; \forall x_i : y_i = -1 \Rightarrow \mathbf{w}^T x_i + w_0 \leq -1$$

and the term to maximize can be inverted in order to get

$$\min \frac{||\mathbf{w}||}{2} = \frac{\sqrt{w^T w}}{2}$$

which is not a quadratic function but it's monotonic so if we found a maximum it will be the same even squared so we can minimize doing $\min \frac{||\mathbf{w}||^2}{2}$ and to summarize the constraints we can just say $yf(x) \geq 1$ since it's our confidence

## Margin Error Bound (just a citation not study material)

**Margin Error Bound**: $\nu + \sqrt{\dfrac{c}{m}(\dfrac{R^2 \bigwedge^2}{\mathbb{R}^2} \ln^2 m + \ln(\dfrac{1}{\delta}))}$

The probability of test error so depends on:

- $\nu$ is number of margin errors (samples that are outside the confidence margin, correcly classified samples with low confidence)

- $m$ training example in the $\sqrt{\dfrac{\ln^2 m}{m}}$ so the result goes down if $m$ goes up
- $R$ is the radius of the space containing all the samples
- larger the margin $\mathbb{R}$, the smaller test error (so we want the margin $\dfrac{2}{\|\mathbf{w}\|}$ to be large)

  if $\mathbb{R}$ is fixed to 1, maximizing margin corresponds to minimizing $||\mathbf{w}||$
- $c$ is a constant

  it makes an upper bound of the generalization error (?)

The name **hard margin** is because we require all examples to be at confidence margin at least one.

## Learning Problem

The learning problem is formalized like $\min \frac{||\mathbf{w}||^2}{2}$ with linear constraints in w $y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1, \forall(\mathbf{x}_i, y_i) \in D$. Still this is a quadratic optimization problem which means that is convex and it has only one global optimum. Problem now is that we need to minimize respect to the constraints and one way to do this is the **KKT approach**

### Karush-Kuhn-Tucker (KKT) approach

With this approach basically we turn a *constrained problem* into an *uncostrained* one with the same solution. To do this suppose we have $f(z)$ to minimize with some constraints like $g_i(z) \geq 0 \forall \mathbf{i}$. Now how can we het rid of the constraints? To do so we introduce a non negative variable called **Lagrange multiplier** noted with $\alpha_i \geq 0$ for each constraint and we rewrite the optimization problem as a **Lagrangian**:

$$\min_z \max_{\alpha \geq 0} f(z) - \sum_i \alpha_i g_i(z)$$

If we find an optimum of this lagrangian called $z^*$ it's still an optimum for the original constrained problem. That's because suppose we find a solution called $z'$ than:

- if at least one constraint is not satisfied ($\exists i \mid g_i(z') < 0$), maximizing over $\alpha_i$ leads to an infinite value;
- if all constraints are satisfied, maximizing over $\alpha$ sets all elements in the sum to zero so that $z'$ is a solution for $\min_z f(z)$.

Applying the approach to our learning problem we will get that

$$\min_{\mathbf{w},w_0} \frac{1}{2}||\mathbf{w}||^2$$

subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1$$
$$\forall(\mathbf{x}_i, \mathbf{y_i}) \in D$$

where :

- $z$ will be our $\mathbf{w}$ so $f(z)$ will be $\frac{1}{2}||\mathbf{w}||^2$
- the constraint will be $g_i(z) \geq 0$ so we need to turn $y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1$ into the $g_i$ which will become $y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1 \geq 0$

By substituiting the new obtained terms we get

$$L(\mathbf{w}, w_0, \alpha) = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{m} \alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1)$$

m = |D|

This lagrangian should be ninimized with respect to $\mathbf{w}$ and $w_0$ and maximized with respect to $\alpha_i$. The solution is called saddle point and it's located where ther is the solution for both problems

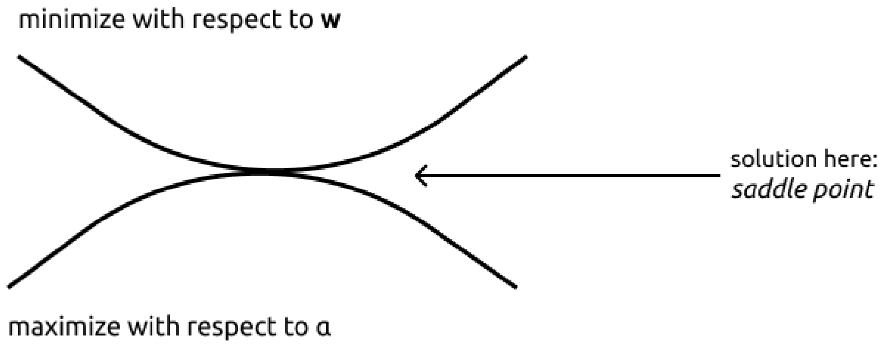
minimize with respect to **w**
solution here: *saddle point*
maximize with respect to α

Figure 1: MLClassifier.png

Going further with the calculus we get that

$$L(\mathbf{w}, w_0, \alpha) = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{m} \alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1)$$

which is our lagrangian. Now we want to minimize with respect to $\mathbf{w}, w_0$ and maximize with respect to $\alpha$. We are interested in computing the gradient respect to our **primal variables w** and $w_0$

So we take $\nabla_{\mathbf{w}} L = \nabla_{\mathbf{w}} \frac{\mathbf{w}^T\mathbf{w}}{2} - \nabla_{\mathbf{w}} \sum_i \alpha_i y_i \mathbf{w}^T\mathbf{x} = \frac{\cancel{2}\mathbf{w}}{\cancel{2}} - \sum_i \alpha_i y_i \mathbf{x}$

and now we set $\mathbf{w} - \sum_i \alpha_i y_i \mathbf{x} = 0$ getting $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}$, but this is not the solution because $\mathbf{w}$ is defined in terms of $\alpha$

Than we can also take the derivative in order to get

$$\frac{\delta L}{\delta w_0} = \frac{\delta(-\sum_i \alpha_i y_i w_0)}{\delta w_0} = -\sum_i \alpha_i y_i = 0 \rightarrow \sum_i \alpha_i y_i = 0$$

---

CERCASI ANIMA PIA CHE TRASCRIVI LA FORMULA LEZ 18.6 DAL MIN 46:00 O DALLE SLIDE

---

After all the steps we get a function which is only a function of the dual variables (the alphas) without our primal variable which is like

$$-\frac{1}{2}\sum_i \sum_j \alpha_i y_i \alpha_j y_j x_i^T x_j + \sum \alpha_i$$

which needs to be maximized respect to $\alpha$ with constraints of

$$\alpha_i \geq 0 \quad \forall i$$
$$\sum_i \alpha_i y_i = 0$$

But still this result is a quadratic optimization problem respect to alpha. In all of this we can see that the beforementioned **primal variables** are missing and replaced with a new pair of variables which will be called **dual variables**(the alphas). This new type of formalization is called **dual formulation**

The result is that $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ can be written both in form of the primal and of the dual because we know that w is equal to $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$

## Decision fucntion

When we did the gradient with respect to **w** previously we got $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}$.

Now if we plug it into our f(x) we get that

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + w_0$$

The decision $f(\mathbf{x})$ (defined as **decision function**) on $\mathbf{x}$ is basically taken as a linear combination of dot products between training points and $\mathbf{x}$, so if $\mathbf{x}_i$ is similar to $\mathbf{x}$ it will have a high dot product because here the dot product works kind of a similarity between the points. Plus the weights of the combination are $\alpha_i y_i$ where large $\alpha_i$ implies large contribution toward class $y_i$

## KKT conditions

To understand wheter a training examples contributes or not to our decision function can be found by appliying KKT. The formulation remains the same so:

$$L(\mathbf{w}, w_0, \alpha) = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{m} \alpha_i (y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1)$$

In the optimal solution $\alpha_i(y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1)$ should be $= 0$, either:

- $\alpha_i = 0$, so the example $\mathbf{x}_i$ does not contribute to the final solution
- if $\alpha_i > 0$ than $y_i(\mathbf{w}^T \mathbf{x}_i + w_0) = 1$, so the confidence for the example should be 1

Graphically we will achieve something like this:

In the image the whited out examples are the ones with $\alpha_i = 0$ and they do not contribute to the decision. The examples which contributes to the whole examples are the one in the dotted lines and they have $\alpha_i > 0$ and are examples for which $y_i(\mathbf{w}^T \mathbf{x}_i + w_0) = 1$. Still the dotted lines are the confidence 1 hyperplanes. These minimal confidence hyperplanes called **support vectors**. All others do not contribute in any way to our decision. SVM are *sparse* which means that they have few support vectors.

Now if we take a step back we know that our $f(x)$ is formed as $w^T x + w_0$. While we found how to compute the first term we still to resolve how to compute the bias

### KKT bias

To compute the bias we can still use the KKT solution we reached before. Given the KKT has found an optimal solution we know that $y_i(\mathbf{w}^T \mathbf{x}_i + w_0) = 1$ must

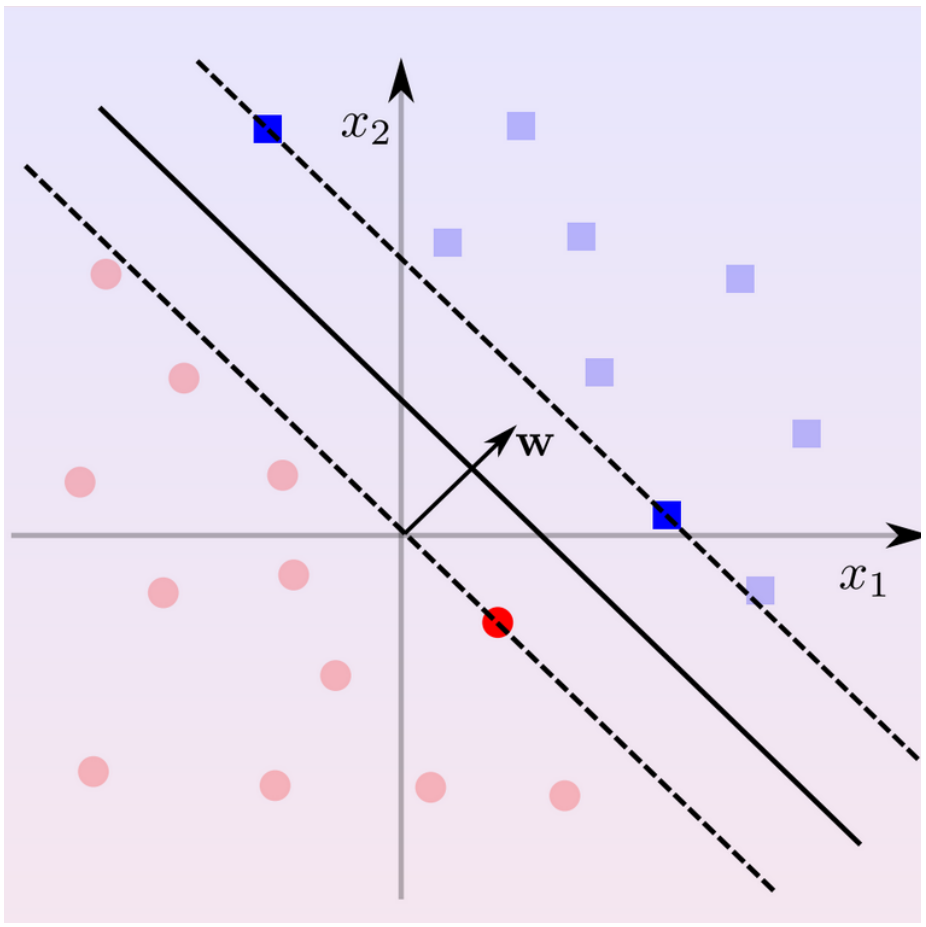

Figure 2: SVM.png

be complied. Fact is we know already that this condition is satisfied so we can just resolve the equation computing for $w_0$ where

$$y_i(\mathbf{w}^T\mathbf{x}_i + w_0) = 1 y_i\mathbf{w}^T\mathbf{x}_i + y_i w_0 = 1 w_0 = \frac{1 - y_i w^T x_i}{y_i}$$

Usually for numerical robustness we compute the bias over all support vectors

## Soft Margin

Until now we just talked about Hard Margin but it has some problems. When we learn a model with the hard margin it could happen that the confidence margin can be too narrow since we are just optimizing the problem to find a solution. With **soft margin** opposite to the hard, we tolerate some errors in order to reach a wide margin. To obtain this we introduce something called **slack variable** in the constraints in order to reach something like:

$$\begin{aligned} \min_{\mathbf{w},w_0} \quad & \frac{||\mathbf{w}||^2}{2} \\ \text{s.t.} \quad & y_i(\mathbf{w}^T x_i + w_0) \geq 1 - \xi_i \end{aligned}$$

The $\xi_i$ will bw our **slack variables** which must be $\geq 0$ where:

- if $\xi_i$ will be equal to 0 than the constraint should be hard satisfied;
- if $\xi_i$ will be greater than 0 than the confidence will be lower than 1 depending on the value of the slack variable itself.

Since we want to pay the use of this variable we add a sum to the term which will be minimized (ideally want the sum of $\xi_i$ to be as low as possible since we are trying to minimize) to get

$$\min_{\mathbf{w},w_0} \quad \frac{||\mathbf{w}||^2}{2} + \sum_i \xi_i$$

Now the min term it's a combination of two components we introduce a coefficient called **regularization term** ($C$ which is a hyperparameter) to give a measure of tradeoff between them. In particular the first component will be the one related to the **margin** while the second will be the **fitting of the training set** meaning how much the training examples satisfy the confidence 1 requirement.

$$\min_{\mathbf{w},w_0} \quad \frac{||\mathbf{w}||^2}{2} + C\sum_i \xi_i$$

The full formalization will be like

$$\begin{aligned}
\min_{\mathbf{w}\in X, w_0\in\mathbb{R}, \xi\in\mathbb{R}^m} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\
\text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i, \quad i = 1, ..., m \\
\xi_i \geq 0, \quad & i = 1, ..., m
\end{aligned}$$

## Regularization theory

The concept of having a margin and a fitting of the training data is an instance of a more genral framework for statistical learning which is called **regularization theory** which basically suggests that we should combine when we learn, a component which penalizes complex solution or encourages generalization with a term that encourages fitting of the training examples

$$\min_{\mathbf{w}\in X, w_0\in\mathbb{R}, \xi\in\mathbb{R}^m} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} l(y_i, f(\mathbf{x}_i))$$

In this case $l$ is a loss function that tells how much we pay for a missclassification and is basically our slack variables

### Hinge loss

Now that we know that our slack variable is the loss function we can establish that $\xi_i = l(y_i, f(\mathbf{x}_i))$. Rewriting the possible values $\xi_i$ can have we get that

$$\xi_i = \begin{cases} 0 \quad \text{if} \quad y_i f(x_i) \geq 1 \\ 1 - y_i f(x_i) \geq 1 \quad \text{otherwise} \end{cases}$$

we get that

$$l(y_i, f(\mathbf{x}_i)) = |1 - y_i f(\mathbf{x}_i)|_+ = |1 - y_i(\mathbf{w}^T\mathbf{x}_i + w_0)|_+$$

The plus means that the value must be 0 if the content is negative. Through this formalization we zero every value of $yf(x)$ if it is greater than 1. This function is called **hinge loss** and it's made in order to:

- where the confidence is more than 1 we don't pay anything
- where is lower tha 1 we pay linearly

**Set hyperparameter C** Still we need to set our C. To do this we do the **cross validation procedure** where we have our training set and the supervised examples called validation set and we evaluate subsets of example to test our model. We will train with different C in order to get the best results

## Dual Regularization formulation

We already said that for optimization problems we can resort to the Lagrangian with the hard margin case. The same thing could be done for soft margins addisng some tweaks where it will result like

$$L = C\sum_i \xi_i + \frac{1}{2}||\mathbf{w}||^2 - \sum_i \alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1 + \xi_i) - \sum_i \beta_i\xi_i$$

$\beta_i$ is the **Lagrange multiplier** for the constraint $\xi_i \geq 0$. Now we need to semplify this by doing the gradient

$$\nabla_w L = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i \to \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

we can see that until now nothing changes form the previous formulation for the hard margin. Also for the derivative of $w_0$ nothing changes ($\frac{\delta L}{\delta w_0} \to \sum_i \alpha_i y_i = 0$)

But with respect to the hard margin we got a new set of primal variables which are the slack variables. We want to minimize even for them so we can take the derivative of the lagrangian wrt the slack variables

$$\frac{\delta L}{\delta \xi_i} = 0 \to C - \alpha_i - \beta_i = 0$$

In the end, substituting in the *Lagrangian* we get: $L(\alpha) = \sum_{i-1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j+1}^{m} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$

So the **dual formulation** is

$$\begin{aligned} \max_{\alpha\in\mathbb{R}^m} \quad & \sum_{i-1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \\ \text{subject to:} \quad & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

**FULL DUAL FORMULATION**

$0 \leq \alpha_i \leq C$ is a derivation from different constraints which are $\alpha_i \geq 0, \beta_i \geq 0, C - \alpha_i - \beta_i = 0$

Different from hard margin SVM because $\alpha$ is upper-bounded by $C$

**Support vector soft margin**

In the Lagrangian, when we get to the saddle point, the result $\forall i$ is

$$\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1 + \xi_i = 0\beta_i\xi_i = 0$$

thus, *support vectors* (where $\alpha_i > 0$) are examples for which is $y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \leq 1$ otherwise our $\alpha$ would be equal to 0. So the support vectors are the ones where the confidence is one or less depending on whether $\xi$ is equal to 0 or not. If:

- $\xi = 0$ than $y_i f(x_i) = 1$
- otherwise smaller

When $\alpha_i < C$, than due to the constraint $C - \alpha_i - \beta_i = 0$ our $\beta$ must be $\beta_i > 0$ and then to hold $\beta_i\xi_i = 0$ our $\xi_i$ must be $\xi_i = 0$ . These support vectors are called **unbound SV**, because they stay in the confidence = 1 hyperplane ($y_i f(\mathbf{x}_i) = y_i(\mathbf{w}^T\mathbf{x}_i + w_0) = 1$)

If $\alpha_i = C$ then our $\xi_i$ can be greater than zero. The support vector generated with this particular $\alpha$ are called **bound SV** and in such case the SV are *margin errors* since they stay on the other part of our confidence 1 area

Margin errors can be also *training errors* if they are in the wrong side of the hyperplane.

Aggiungere immagine min 26:00 svm part 3

# Large Scale SVM (Pegasus)

Training of SVM is a quadratic optimization problem; if the dataset is large, to train more quickly there is the **stochastic gradient descent**.

Our learning objective should be something like $\frac{||\mathbf{w}||^2}{2} + C\sum_i l(y_i, f(x_i))$ and the loss of the SVM (hinge loss) is $l(y_i, f(x_i)) = |1 - y_i f(x_i)|_+$. If we plug it on our learning objective , even if it is not completely smooth, we get

$$\frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{m}|1 - y_i(\mathbf{w}^T\mathbf{x}_i + w_0)|_+$$

Than instead of having a C what happens if u have a lot of examples is that the total loss depends even on the number of examples so we divide it by the number of the examples themselves in order to get $C = \frac{1}{m}$. In order to have full fidelity to the original algorithm there is a $\lambda$ which basing on recording is not very important idk so the final result would be

$$\min_{\mathbf{w}\in X} \frac{\lambda\|\mathbf{w}\|^2}{2} + \frac{1}{m}\sum_{i=1}^{m} |1 - y_i\langle \mathbf{w}, \mathbf{x}_i \mathbb{R}angle|_+$$

Now we want to do the stochastic gradient descent. To do this (recap only) we take the error function for the single example, we compute the gradient , then we update it and we move to the next example. So at first we have

$$E(\mathbf{w}; (x_i, y_i)) = \frac{\lambda}{2}||\mathbf{w}||^2 + |1 - y_i\langle \mathbf{w}, \mathbf{x}_i \mathbb{R}angle|_+$$

> Here we dont care about bias since it's not very relevant due to the number of features

Now we want to compute the gradient but if we remember the hinge loss function it is not derivable. So to do this we compute the **subgradient**.

> The **subgradient** of a function $f$ at a point $\mathbf{x}_0$ is any vector $\mathbf{v}$ such that for any $\mathbf{x}$ that this holds: $f(\mathbf{x}) - f(\mathbf{x}_0) \geq \mathbf{v}^T(\mathbf{x} - \mathbf{x}_0)$, it means you can use any of this vector as gradient in points where the derivatives doesn't exists. His function is to find some gradients in a non derivability point

The way of formally doing this is to apply an indicator function like

$$\not\Vdash[y_1\langle \mathbf{w}, \mathbf{x}_i \mathbb{R}angle < 1] = \begin{cases} 1 & \text{if } y_i\langle \mathbf{w}, \mathbf{x}_i \mathbb{R}angle < 1 \\ 0 & \text{otherwise} \end{cases}$$

and computing the gradient in the part where $y_i\langle \mathbf{w}, \mathbf{x}_i \mathbb{R}angle < 1$ gives as a result $y_i x_i$. So plugging it in our gradient step we get

$$\nabla_{\mathbf{x}} E(\mathbf{w}, (\mathbf{x}_i, y_i)) = \lambda\mathbf{w} - \not\Vdash[y_1\langle \mathbf{w}, \mathbf{x}_i \mathbb{R}angle < 1] y_i \mathbf{x}_i$$

where if the confidence is lower than one than we will apply $x_i y_i$ otherwise it will be 0.

## Pegasus Algorithm

The algorithm to do the large scale learning is called *Pegasus* which goes like:

1. Initialize $\mathbf{w}_1 = 0$
2. for $t = 1$ to $T$:
    1. Randomly choose $(\mathbf{x}_{i_t}, y_{i_t})$ from $D$
    2. Set $\eta_t = \frac{1}{\lambda t}$

    3. Update $\mathbf{w}$:
$$w_{t+1} = \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} E(\mathbf{w}, (\mathbf{x}_{i_t}, y_{i_t}))$$

3. Return $\mathbf{w}_{T+1}$

The *learning rate* is not static, it's an **adaptive learning rate** that decreases with $t$: The choice of the learning rate allows to bound the runtime for an $\epsilon$-accurate solution to $\mathbb{O}(\frac{d}{\lambda\epsilon})$ with $d$ maximum number of non-zero features in an example (guarantees accuracy of solution)