Machine Learning Algebra

By Matthieu Lagarde

April 9, 2022

1 Multivariate linear regression

1.1 Notations

Let m be the number of examples or observations in our training set. Let n be the number of features or explanatory variables observed for each example of the training set. Let $x_j^{(i)}$ be the value of feature j for example i. Let $y^{(i)}$ be the output value for example i.

$$y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{pmatrix} \in \mathbb{R}^m \tag{1}$$

y is called the output vector. It contains the output values of the training set.

$$X = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \dots & \dots & \dots & \dots & \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \in \mathbb{R}^{m\times(n+1)} \tag{2}$$

X is called the data matrix. If we ignore the first column of 1, each row of matrix X is one example of the training set and each column of the matrix X is the values observed for one feature in the training set.

$$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{pmatrix} \in \mathbb{R}^{(n+1)} \tag{3}$$

θ is the vector of parameters.

1.2 Hypothesis

We assume that there is a linear relationship between the features and the output. Note that the relationship is linear in θ but the features themselves can be non linear transformations of the initial features such as quadratic terms or interaction terms.

The unvectorized form of the hypothesis for a given example i is:

$$h_\theta(x^{(i)}) = \sum_{j=0}^{n} \theta_j * x_j^{(i)} \in \mathbb{R} \; with \; x_0^{(i)} = 1 \tag{4}$$

The vectorized form of the hypothesis can be written as follows:

$$h_\theta(X) = X\theta \in \mathbb{R}^m \tag{5}$$

For a single example, we can also write the vectorized form of the hypothesis:

$$x^{(i)} = \begin{pmatrix} 1 \\ x_1^{(i)} \\ \dots \\ x_n^{(i)} \end{pmatrix} \in \mathbb{R}^{(n+1)}$$

$$h_\theta(x^{(i)}) = {x^{(i)}}^\top \theta \in \mathbb{R}$$

1.3 Cost function

The unvectorized form of the cost function is:

$$\begin{aligned} &J(\theta) = \frac{1}{2m} * \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \in \mathbb{R} \\ &where \; x^{(i)} \in \mathbb{R}^{(n+1)} \; and \\ &where \; y^{(i)} \in \mathbb{R} \end{aligned}$$

The vectorized form of the cost function is:

$$J(\theta) = \frac{1}{2m} * (X\theta - y)^\top (X\theta - y) \in \mathbb{R} \tag{6}$$

1.4 Normal equation

$J(\theta)$ is convex so any local minimum is a global minimum. We thus know that θ^* obeys the following equation:

$$\nabla J(\theta^*) = 0_{(n+1)}$$

Let recall a property of matrix differentiation. Let α be a scalar equal to $y^\top x$ where y and x be two column vectors of $\mathbb{R}^m$ that are respectively a function of another column vector z of $\mathbb{R}^n$. We have:

$$\alpha = y^\top x$$

$$\frac{\partial \alpha}{\partial z} = \frac{\partial y}{\partial z}^\top x + \frac{\partial x}{\partial z}^\top y \in \mathbb{R}^n$$

Using this property, we have:

$$\begin{aligned}
&1/2m * 2 * \frac{\partial (X\theta^* - y)}{\partial \theta}^\top (X\theta^* - y) = 0_{(n+1)} \\
&\iff 1/m * X^\top (X\theta^* - y) = 0_{(n+1)} \\
&\iff X^\top X\theta^* - X^\top y = 0_{(n+1)} \\
&\boxed{\iff \theta^* = (X^\top X)^{-1} X^\top y \in \mathbb{R}^{(n+1)}}
\end{aligned}$$

1.5 Gradient descent

If necessary, here is the vectorized implementation of gradient descent. Denoting α the learning rate:

$$\theta := \theta - \frac{\alpha}{m} * X^\top (X\theta - y) \in \mathbb{R}^{(n+1)} \tag{7}$$

1.6 Regularized cost function

Denoting λ the regularization parameter, the unvectorized form of the cost function can be written as:

$$J(\theta) = \frac{1}{2m} * \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} * \sum_{j=1}^{n} \theta_j^2 \in \mathbb{R}$$
$$where\ x^{(i)} \in \mathbb{R}^{(n+1)}\ and$$
$$where\ y^{(i)} \in \mathbb{R}$$

Be careful, in the regularization part of the expression (the second sum), the j index goes from 1 to n and NOT from 0 to n. Indeed, by convention, we do not regularize θ_0.

The vectorized form of the regularized cost function is thus:

$$J(\theta) = \frac{1}{2m} * (X\theta - y)^\top (X\theta - y) + \frac{\lambda}{2m} * \theta_r^\top \theta_r$$
$$where\ \theta_r = \begin{pmatrix} \theta_1 \\ ... \\ \theta_n \end{pmatrix} \in \mathbb{R}^n$$

1.7 Regularized normal equation

The vectorized form of the regularized normal equation is:

$$\theta^* = (X^\top X + \lambda * M)^{-1} X^\top y \in \mathbb{R}^{(n+1)}$$
$$where\ M = \begin{pmatrix} 0 & 0 & 0 & ... & 0 \\ 0 & 1 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & 1 & 0 \\ 0 & 0 & ... & 0 & 1 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$$

1.8 Regularized gradient descent

Let compute the gradient of $J(\theta)$ when it is regularized. There are two cases, one for the partial derivative with respect to θ_0 and one for the partial derivatives with respect to θ_j:

$$\frac{\partial J(\theta)}{\partial \theta_0} = \left[\frac{1}{m} * X^\top (X\theta - y)\right]_1 \; i.e.\ 1st\ element\ of\ previous\ gradient$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left[\frac{1}{m} * X^\top (X\theta - y) + \frac{\lambda}{m} * \theta\right]_{j+1} \; for\ j \in \{1, 2, ..., n\}$$

2 Logistic regression

2.1 Sigmoid function

Let introduce the sigmoid function:

$$g \ : \ x \in \mathbb{R} \longrightarrow \frac{1}{1 + e^{-x}} \in \,]0, 1[\tag{8}$$

The interesting properties of the sigmoid function are:

- It is defined over $\mathbb{R}$.
- It is increasing.
- $g(0) = \frac{1}{2}$.
- It is converging to 0 in $-\infty$ and to 1 in $+\infty$.
- It is convex over $]-\infty, 0]$ and concave over $[0, +\infty[$.
- It means that (0, 0.5) is an inflection point of g.
- $g(-4) = 0.02$ and $g(4) = 0.98$

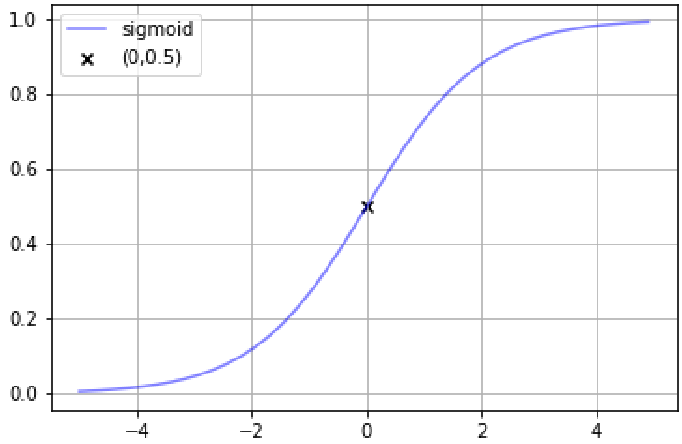

Figure 1: Graph of the sigmoid function

2.2 Hypothesis

The unvectorized form of the hypothesis for a given example i is:

$$h_\theta(x^{(i)}) = g\left(\sum_{j=0}^{n} \theta_j * x_j^{(i)}\right) \in \,]0,1[$$

$$where\ g\ is\ the\ sigmoid\ function.$$

The hypothesis can be interpreted as the probability that a new observation belongs to the positive class i.e. that $y = 1$ given the value of its features i.e. given x, parameterized by θ. In other words:

$$h_\theta(x) = P\left(y = 1 | x; \theta\right) \tag{9}$$

We then introduce the following decision rule:

$$y = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 \\ 0 & \text{if } h_\theta(x) < 0.5 \end{cases} \tag{10}$$

Finally, the vectorized form of the hypothesis can be written as follows:

$h_\theta(X) = g\left(X\theta\right) \in \left]0, 1\right[^m$
where g is the sigmoid function applied element-wise

2.3 Cost function

Before introducing the cost function, let study two functions:

$$v \;:\; x \in \left]0, 1\right[\longrightarrow -ln(x) \in \left[0, +\infty\right[$$

$$w \;:\; x \in \left]0, 1\right[\longrightarrow -ln(1-x) \in \left[0, +\infty\right[$$

Function v has the following properties on the interval $\left]0, 1\right[$:

- It is decreasing.
- It converges to $+\infty$ in 0.
- $v(1) = 0$.

Function w has the following properties on the interval $\left]0, 1\right[$:

- It is increasing.
- It converges to $+\infty$ in 1.
- $w(0) = 0$.

Here is the graph of v and w on the interval $\left]0, 1\right[$:

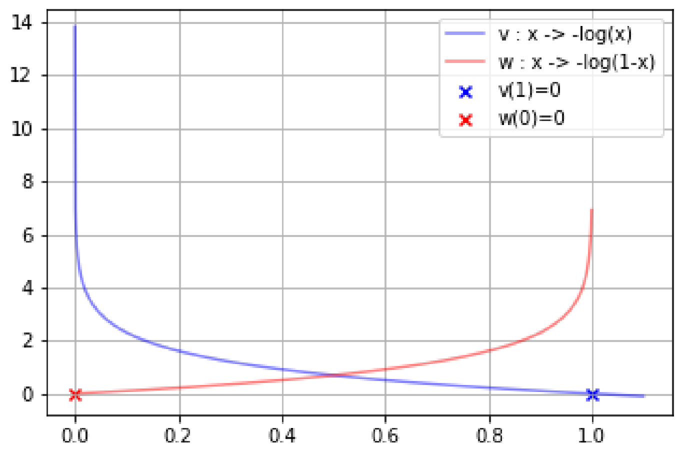

Figure 2: Graph of v and w

Now we introduce the following cost function:

$$J(\theta) = \frac{1}{m} * \sum_{i=1}^{m} Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) \in \mathbb{R}$$

where

$$Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) = \begin{cases} -ln\left(h_\theta\left(x^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -ln\left(1 - h_\theta\left(x^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

Given that $y \in \{0, 1\}$, the cost function can be rewritten as:

$$J(\theta) = \frac{1}{m} * \sum_{i=1}^{m} \left[\left(y^{(i)} * -ln\left(h_\theta\left(x^{(i)}\right)\right)\right) + \left(\left(1 - y^{(i)}\right) * -ln\left(1 - h_\theta\left(x^{(i)}\right)\right)\right)\right] \in \mathbb{R} \tag{11}$$

We can also write the vectorized form of the cost function as follows:

$$J(\theta) = -\frac{1}{m} * \left[y^\top ln(h_\theta(X)) + (1 - y)^\top ln(1 - h_\theta(X))\right] \in \mathbb{R}$$

where ln is applied element-wise

2.4 Gradient descent

The unvectorized form of the gradient of the cost function can be written as:

$$\frac{\partial J(\theta)}{\partial \theta_k} = \frac{1}{m} * \sum_{i=1}^{m} \left(\frac{1}{1 + exp(-\sum_{j=0}^{n} x_j^{(i)} * \theta_j)} - y^{(i)} \right) * x_k^{(i)} \in \mathbb{R} \tag{12}$$

The vectorized form of the gradient of the cost function can be written as:

$$\nabla J(\theta) = \frac{1}{m} * X^\top \left(h_\theta(X) - y\right) \in \mathbb{R}^{n+1}$$

where $h_\theta(X) \in \,]0, 1[^m$

Denoting α the learning rate, a vectorized implementation of the gradient descent can thus be written as:

$$\theta := \theta - \frac{\alpha}{m} * X^\top \left(h_\theta(X) - y\right) \in \mathbb{R}^{n+1}$$

where $h_\theta(X) = g(X\theta) \in \,]0, 1[^m$

and g is the sigmoid function applied element-wise

2.5 Regularization

The regularized cost function can be written as:

$$J(\theta) = -\frac{1}{m} * \left[y^\top ln(h_\theta(X)) + (1 - y)^\top ln(1 - h_\theta(X)) \right] + \frac{\lambda}{2m} * \theta_r^\top \theta_r \in \mathbb{R}$$

where ln is applied element-wise

The regularized gradient can be written as:

$\left[\widetilde{\nabla J(\theta)}\right]_1 = [\nabla J(\theta)]_1$for $\partial\theta_0$ i.e. the first element of the regularized gradient

$\left[\widetilde{\nabla J(\theta)}\right]_j = \left[\nabla J(\theta) + \frac{\lambda}{m} * \theta\right]_j$ where $j \in \{2, 3, ..., n\}$

3 Neural network classifier

3.1 Notations

Let consider the following neural network:

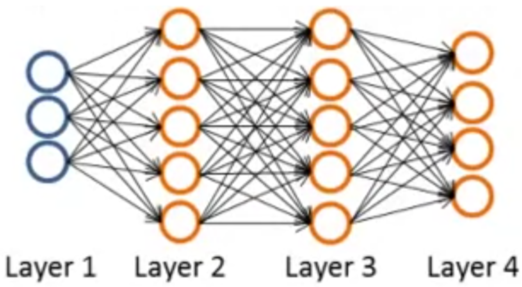

Figure 3: Example of neural network

We assume that we face a multi-class classification where we want to predict whether an observation belongs to one of K possible classes. We denote L the total number of layers in the network. In the example, there are 4 layers. The first layer is called the input layer. The last layer is called the output layer. Intermediate layers are called hidden layers. Here, there are 2 hidden layers. We denote s_l the number of units in layer l, not counting the bias unit. In the example, we thus have:

- $s_1 = 3$ which is n, the number of input features.
- $s_2 = 5$.
- $s_3 = 5$.
- $s_4 = 4$ which is K, the number of classes.

We assume that a bias unit is added as an input to any layer.

We denote m the total number of examples in our training set. We denote $x^{(i)} \in \mathbb{R}^{(s_1+1)}$ the vector of features of the $i-th$ example in the training set where $x_0^{(i)} = 1$. We denote $y^{(i)} \in \mathbb{R}^{K=s_L}$ the output vector of the $i-th$ example in the training set. If the $i-th$ example belongs to class k, then:

$$y^{(i)} = \begin{pmatrix} 0 \\ ... \\ 0 \\ 1 \\ 0 \\ ... \\ 0 \end{pmatrix} \text{ where } \left(y^{(i)}\right)_{k,1} = 1 \in \mathbb{R}^{K=s_L} \tag{13}$$

Note that the full output values can be stored in a matrix:

$$Y = \begin{pmatrix} {y^{(1)}}^\top \\ ... \\ {y^{(m)}}^\top \end{pmatrix} \in \mathbb{R}^{m*K} \tag{14}$$

Finally, we denote $\Theta^{(l)}$ the matrix of parameters required to go from layer l to layer $(l+1)$. This matrix is in the space $\mathbb{R}^{s_{(l+1)}*((s_l)+1)}$ since it maps vectors of dimension $((s_l)+1)$ (do not forget the bias unit) into vectors of dimension $s_{(l+1)}$. Note that:

$$\Theta^{(l)} = \begin{pmatrix} \text{Parameters associated to unit 1 of layer } (l+1) \\ ... \\ \text{Parameters associated to unit } s_{l+1} \text{of layer } (l+1) \end{pmatrix} \in \mathbb{R}^{s_{(l+1)}*((s_l)+1)} \tag{15}$$

In other words, $\left(\Theta^l\right)_{i,j}$ corresponds to the parameter associated to unit j of the layer l, which is used to compute unit i of the layer $(l+1)$. Note that in this example, there are 3 matrices of parameters to estimate : (Θ^1), (Θ^2) and (Θ^3). It implies $5*3+5*5+4*5 = 15+25+20 = 60$ parameters to estimate. In general, there are $(L-1)$ matrices of parameters to estimate.

3.2 Hypothesis

The hypothesis can be written recursively as follows:

$X_1 =$ the data matrix $\in \mathbb{R}^{m*(s_1+1)}$

$X_{l+1} = 1_{m*1} \parallel g(X_l {\Theta^{(l)}}^\top) \in \mathbb{R}^{m*((s_{l+1})+1))}$

$h_\Theta(X_1) = g(X_{L-1} {\Theta^{(L-1)}}^\top) \in \mathbb{R}^{m*s_L}$

where $\parallel$ means horizontal concatenation and g is the sigmoid function applied element-wise.

Again, the hypothesis only gives the probability that an observation belongs to each class. As for the one vs. all multi-class classification problem, we then introduce the following decision rule to pick the most probable class given the input features:

$$\text{Class of } y = index_{of}\,(max_{row\,wise}(h_\Theta(X_1))) \in \mathbb{R}^m \tag{16}$$

3.3 Cost function

Very similar to the cost function for the logistic regression, we just add a cost for the estimation of each output unit (now there are K of them). We also directly introduce the regularization term. Here is an unvectorized expression of the cost function:

$$\begin{aligned} J(\Theta) = -\frac{1}{m} * \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} * ln(h_\Theta(X_1)_{i,k}) + (1 - y_k^{(i)}) * ln(1 - h_\Theta(X_1)_{i,k}) \\ + \frac{\lambda}{2m} * \sum_{l=1}^{L-1} \sum_{i=1}^{s_{(l+1)}} \sum_{j=2}^{(s_l)+1} {\Theta_{i,j}^{(l)}}^2 \end{aligned} \tag{17}$$

It can be vectorized as follows:

$$J(\Theta) = -\frac{1}{m} * \left[\; Tr\left(ln(h_\Theta(X_1))Y^\top\right) + Tr\left(ln(1 - h_\Theta(X_1))(1-Y)^\top\right)\; \right]$$
$$+ \frac{\lambda}{2m} * \sum_{l=1}^{L-1} Tr\left({\Theta_r^{(l)}}^\top \Theta_r^{(l)}\right)$$

where $\Theta_r^{(l)} = \Theta^{(l)}$ without the first column of $\theta_0 s$,
and $h_\Theta(X_1) \in \mathbb{R}^{m*K}$ and $Y \in \mathbb{R}^{m*K}$,
and $Tr(.)$ is the trace of a square matrix operator i.e. the sum of its diagonal,
ln(.) is applied element-wise.

3.4 Backpropagation algorithm

In order to minimize the cost function, we need to compute the partial derivatives of $J(\Theta)$ with respect to all the parameters i.e. with respect to each element of the $(L-1)$ matrices of parameters:

$$\frac{\partial J(\Theta)}{\partial \Theta_{i,j}^{(l)}} \text{ for } i,\ j,\ l \tag{18}$$

To do so, we can use the backpropagation algorithm. This algorithm will yields the partial derivatives of $J(\Theta)$ with respect to each element of the matrices of parameters.

- Step 1: Initialization. Set the matrices of parameters to random small values.
- Step 2: Compute forward the activation of the units of each intermediate layer as well as the hypothesis i.e. the output units with matrices of zeroes.
- Step 3: Compute the error of the last layer.
- Step 4: Compute backward the errors of all previous layers until the first intermediate layer.
- Step 5: Use the errors of each layer to compute the gradients of each matrix of parameters.

On the full sample, the errors can be computed as follows:

$$\delta^{(L)} = A^{(L)} - Y^{(L)} \in \mathbb{R}^{m \times s_L}$$
$$\delta^{(l)} = \delta^{(l+1)}\Theta^{(l)} \;.*\; A^{(l)} .* (1 - A^{(l)}) \in \mathbb{R}^{m \times (s_l+1)}$$
$$\text{where } A^{(l)} = 1_{m*1} \parallel g(\Theta^{(l-1)} X_{l-1}^{\top})$$

Finally, we can obtain the gradients as follows:

$$\Delta^{(l)} = \frac{1}{m} * \delta^{(l+1)^{\top}} A^{(l)} + \frac{\lambda}{m} * \Theta_r^{(l)}$$
$$\text{where } \Theta_r^{(l)} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \end{pmatrix} \parallel \left[\Theta^{(l)}\right]_{col2-end}$$

Note that for $l = 1$, we have $A^1 = X$.

4 Support Vector Machines

4.1 Hypothesis

Support Vector Machines or SVM is another classifier. The hypothesis is a discriminant function based on the value of the linear combination:

$$h_\theta(x^{(i)}) = \begin{cases} 1 & \text{if } \theta^{\top} x^{(i)} >= 0 \\ 0 & \text{otherwise} \end{cases}$$
$$\text{where } x^{(i)} \in \mathbb{R}^{(n+1)} \text{ and } \theta \in \mathbb{R}^{(n+1)}$$

4.2 Cost function

The cost function is inspired by the cost function of the logistic regression but is slightly different:

$$J(\theta) = C * \left[\sum_{i=1}^{m} y^{(i)} * cost_1(\theta^{\top} x^{(i)}) + (1 - y^{(i)}) * cost_0(\theta^{\top} x^{(i)})\right] + \frac{1}{2}\sum_{j=1}^{n} \theta_j^2 \tag{19}$$

Compared to the cost function of the logistic regression:

- We have dropped the $\frac{1}{m}$ factors
- We have replaced the parameter λ by a parameter $C = \frac{1}{\lambda}$. The bigger C, the more we want to fit exactly the data. The lower C, the more we want the parameters to be small.
- The individual cost function assigning a cost to the prediction is the Hinge loss function.

It can be written in a vectorized form as follows:

$$J(\theta) = C * \Big[(Y^\top cost_1(X\theta)) + ((1-Y)^\top cost_0(X\theta))\Big] + \frac{1}{2} * \theta_r^\top \theta_r \in \mathbb{R}$$

where θ_r is equal to θ without θ_0 thus $\in \mathbb{R}^n$

$cost_0(.)$ and $cost_1(.)$are applied element-wise.

4.3 Hinge loss functions

For $y^{(i)} = 1$, the hinge loss function is:

$$cost_1(z) = \begin{cases} 0 & \text{if } z >= 1 \\ k(1-z) & \text{if } z < 1 \end{cases} \text{ with k an arbitrary slope parameter } > 0 \tag{20}$$

For $y^{(i)} = 0$, the hinge loss function is:

$$cost_0(z) = \begin{cases} 0 & \text{if } z < -1 \\ k(1+z) & \text{if } z >= -1 \end{cases} \text{ with k an arbitrary slope parameter } > 0 \tag{21}$$

Here is a graph of $cost_0$ and $cost_1$ on $[-2, 2]$ with $k = 2$:

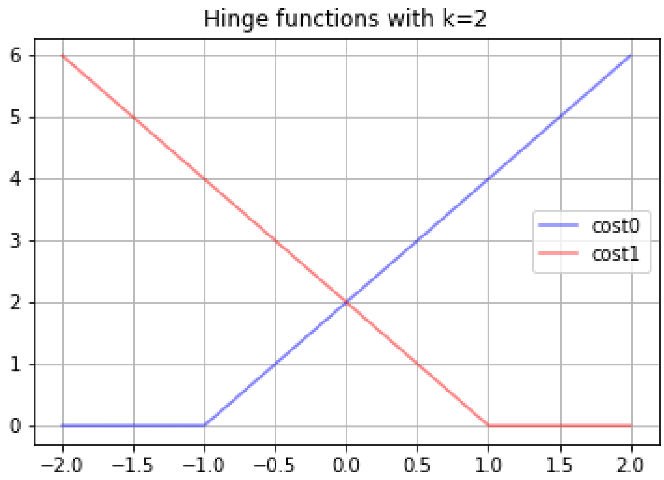

Figure 4: Graph of $cost_0$ and $cost_1$ with $k = 2$

4.4 Large margin intuition

When ...