

Training of Iris flower dataset and comparing performance of different Algorithms

Author – Kornia Bapari

Sections –

- 1 . Libraries used in the code
2. Dataset used in the code
3. Dataset walk Through
4. Dataset through Graphs
5. Build Algorithms
6. Compare different Hypothesis fits of different algos

Section 1 –Libraries used

PANDAS

MATPLOTLIB

SKLEARN

NUMPY

Note : - you can install them using `pip install <library name>`

Section 2 – Dataset used in code

Iris flower dataset

Download from here –

<https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv>

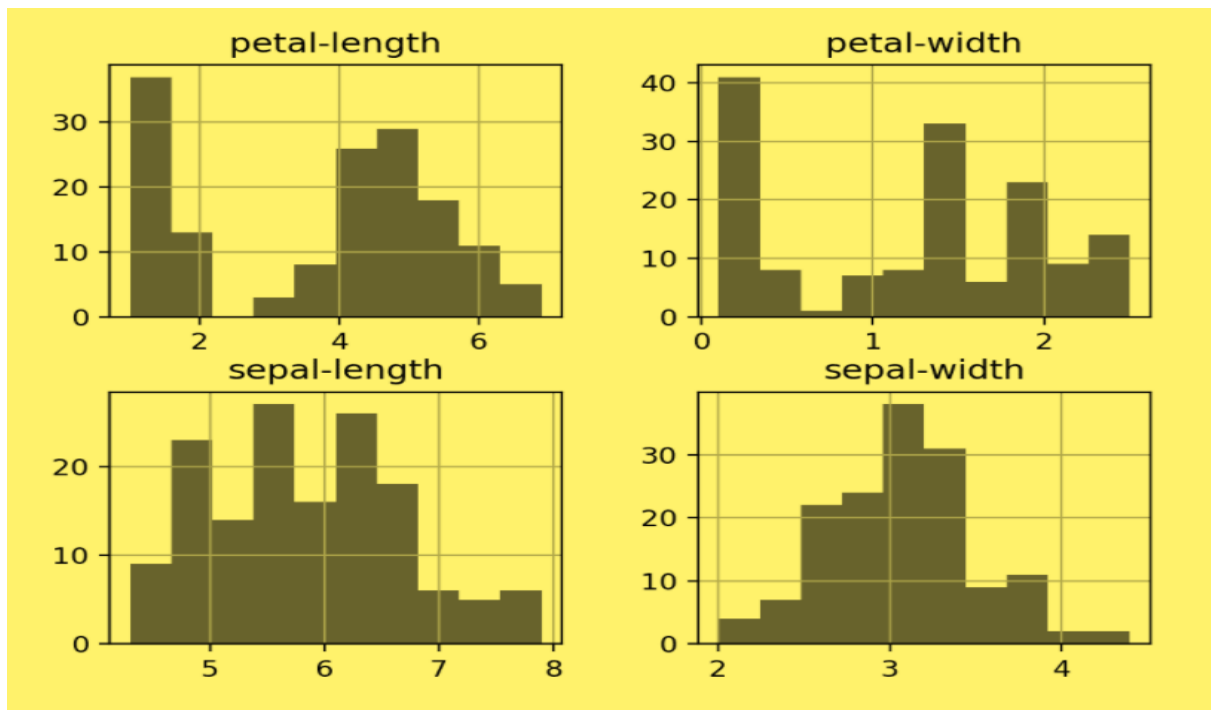
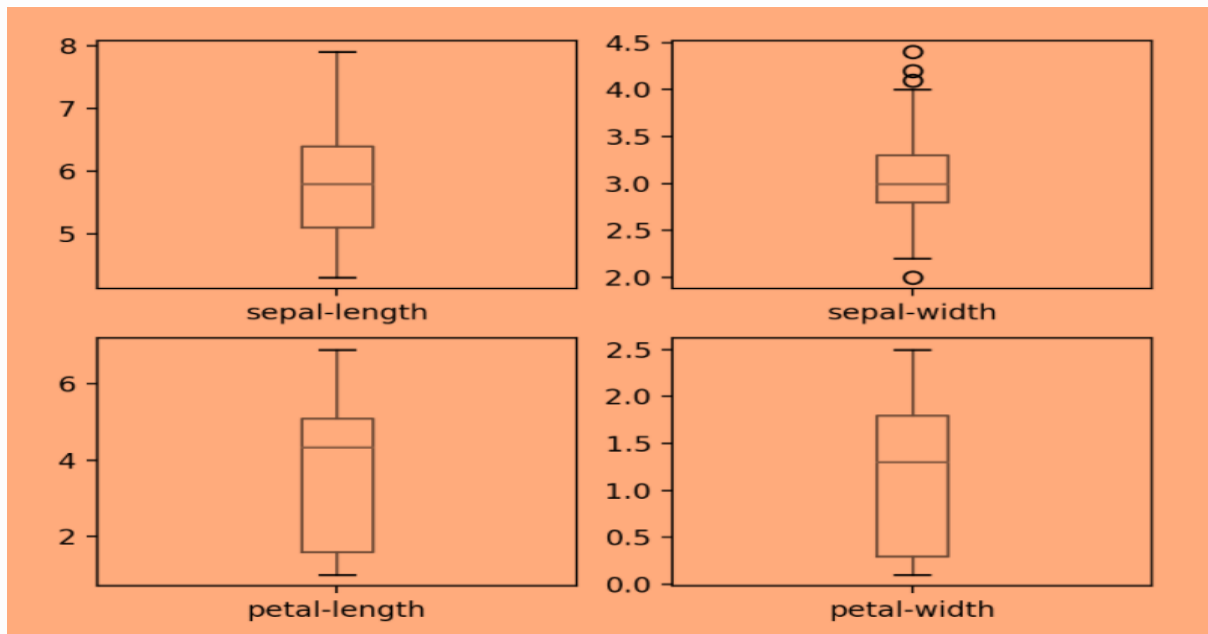
Section 3 - Dataset walk Through

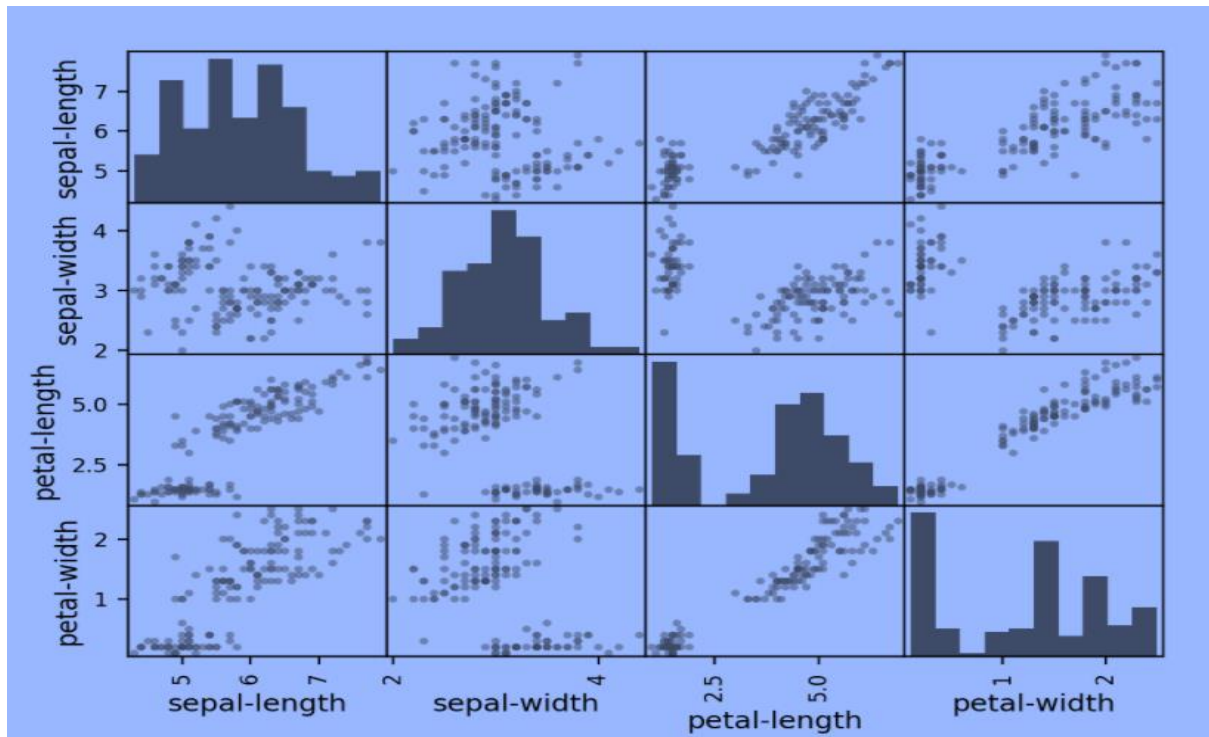
In the imported IRIS dataset we have $m = 150$, and no of features = 5, so $x=5$

Feature Scaling is not required as all features are of same scale-

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Section – 4 – Dataset through Graphs





Section 5 – Build Algorithms

MY own Notes on Logistic regression-

Learning from – Stanford ML course by Andrew NG

Logistic regression

Sigmoid or logistic function

So here we define our hypothesis in term so $\theta^T x$, but instead so just $\theta^T x$ as in linear regression, we here make a little more involved function that we call as sigmoid function as it is given as in below slide, so our hypothesis is thus outputting between zero and 1 due to use of this sigmoid function and we treat this output as probability, thus if our hypothesis outputs 1 we say we 100 probability that tumor is benign

Logistic Regression Model

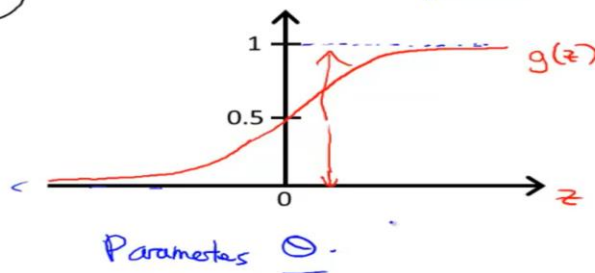
Want $0 \leq h_{\theta}(x) \leq 1$

$$h_{\theta}(x) = g(\theta^T x)$$

$$\rightarrow g(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid function
Logistic function

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$



Andrew Ng

As in below example we chose x as 1 and tumor size, and our hypothesis gives value that we treat as probability

Interpretation of Hypothesis Output

$h_{\theta}(x)$ = estimated probability that $y = 1$ on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$$h_{\theta}(x) = 0.7 \quad y = 1$$

Tell patient that 70% chance of tumor being malignant

$$h_{\theta}(x) = P(y=1|x;\theta)$$

$y = 0$ or 1

"probability that $y = 1$, given x , parameterized by θ "

$$\rightarrow P(y=0|\theta) + P(y=1|\theta) = 1$$

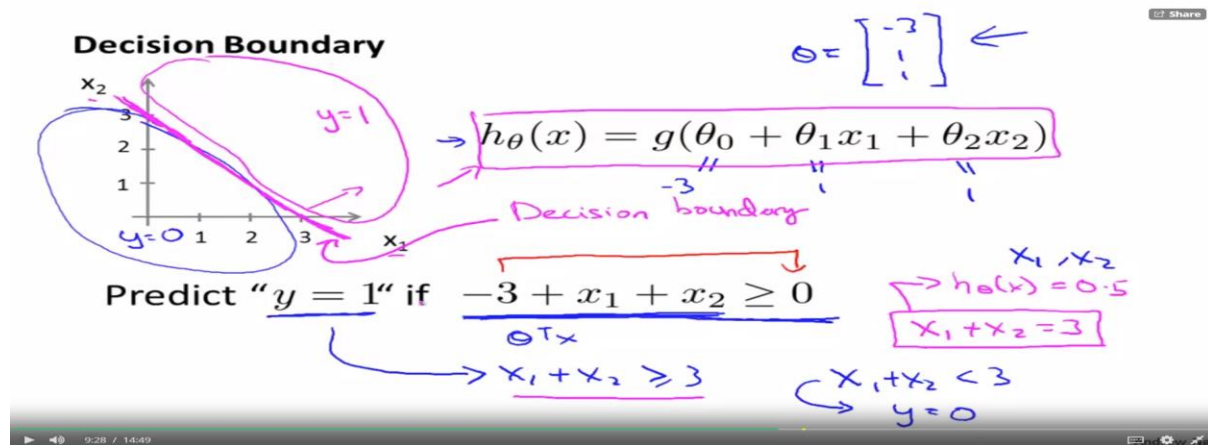
$$\rightarrow P(y=0|x;\theta) = 1 - P(y=1|x;\theta)$$

DECISION BOUNDARIES

These are key terms, these are used to demarcate boundaries from where you can say y is 1 or zero,

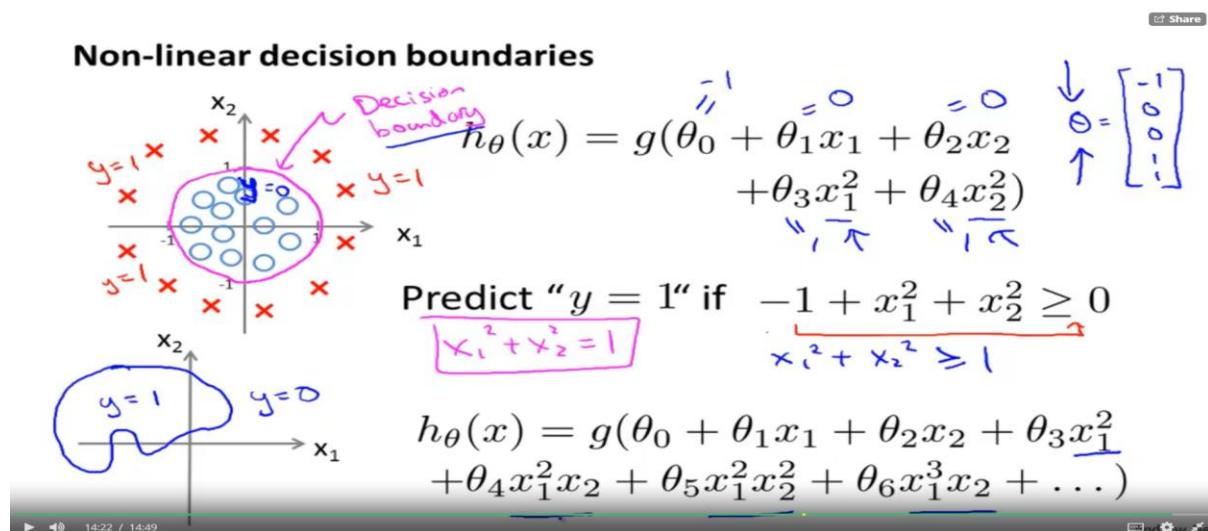
Also if you see sigmoid function, the value of that function is greater than 0.5 when $z > 0$ and z is positive when $\theta^T x$ is positive, so whole problem now boils down to the fact that if $\theta^T x$ is +ve then we take $y=1$ and if negative then $y=0$, so a wonderful way to find the solution to classification problem using sigmoid function property of lying between 0 to 1 and having clear demarcation point of $y=0.5$ at $z=0$

You can see in below example , if $x_1 + x_2 > 3$ then we take $y=1$ otherwise $y=0$

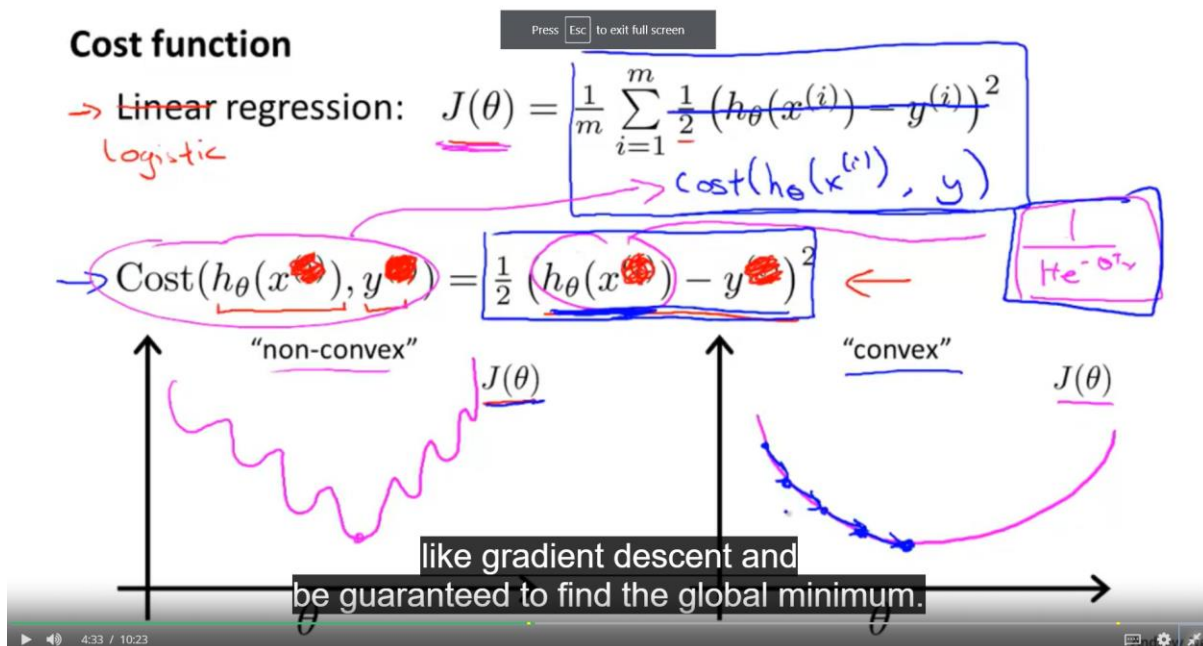


More complex clusters can be classified if you make features like below , so x_1^2 and x_2^2 are features, thus using these features gives power to cluster things inside and outside circles

Also if you use features like $x^2 * y^2$,you can even classify clusters of random shape, so in a way you can see galaxies can be clustered using similar algo, the features might be frequency of light coming .



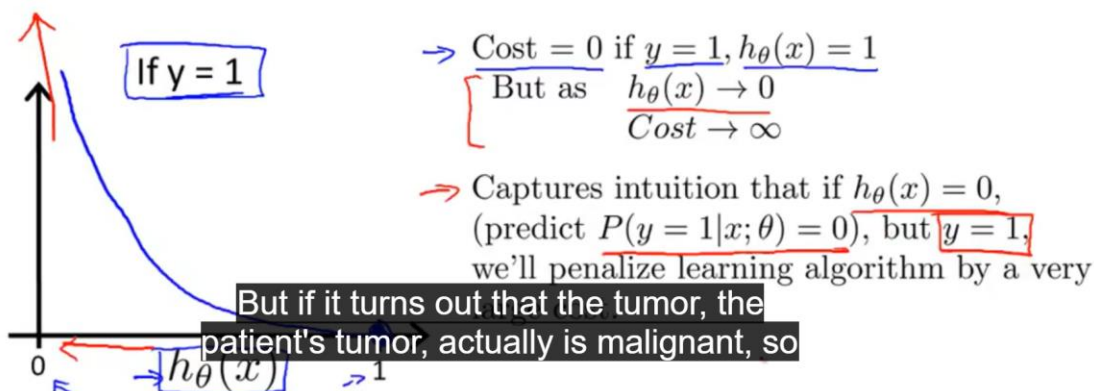
How to choose cost function for a logistic regression problem



Now you can choose a cost function something like we have chosen in linear regression, but it turns out if use use that square valued function here , you get in to a function which is non convex in nature , i.e it might have many local minimas, so we have to pick a cost function that is convex in nature and ensures global minima

Logistic regression cost function

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$



So we pick above cost function , now our choice of hypothesis ensures that we get values between 0 and 1 , so $h(x)$ is always less 1, so we define our cost function as $c = -\log(h(x))$ when $y=1$, remember y is coming from our training set, and $h(x)$ is our hypothesis , so if you see $-\log(h(x))$ you get 0 if $h(x) = 1$, that means cost is 0 if your hypothesis predicts $h(x) = 1$, so nice and if hypothesis predicts 0 but $y = 1$ then cost is infinite , so we see our cost function working intuitively.

Applying gradient descent on logistic regression

Writing cost function in on line

Logistic regression cost function

$$\rightarrow J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$

$$\rightarrow \text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

$$\rightarrow \text{Cost}(h_{\theta}(x), y) = -y \log(h_{\theta}(x)) - (1-y) \log(1 - h_{\theta}(x))$$

If $y=1$: $\text{Cost}(h_{\theta}(x), y) = -\log h_{\theta}(x)$

If $y=0$: $\text{Cost}(h_{\theta}(x), y) = -\log(1 - h_{\theta}(x))$

of taking both of these expressions,
the cases $y=1$ and $y=0$,

Andrew Ng

So our cost function looks like below, we train over all the set so summation sign and we want to minimise the cost of it, so all cool

Logistic regression cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right]$$

To fit parameters θ :

$$\min_{\theta} J(\theta) \quad \text{Get } \theta$$

To make a prediction given new x :

Output $h_{\theta}(x)$ And just to remind you, the output of my hypothesis I'm going to interpret as

Gradient descent, so same formula of parameter updating is applied in logistic regression as linear regression, what's different between the 2 is choice of hypothesis

You can derive for $d/d\theta$ of current cost function it comes same as linear regression

Gradient Descent

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right]$$

Want $\min_{\theta} J(\theta)$:

Repeat {

$$\rightarrow \theta_j := \theta_j - \alpha \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$h_{\theta}(x) = \Theta^T x$$

$$h_{\theta}(x) = \frac{1}{1 + e^{-\Theta^T x}}$$

Algorithm looks identical to linear regression!
that it is converging.

⏪ ⏩ ⏴ ⏵

Andrew Ng

SVM algo –

Support Vector Machine

One of the most used algos in the world, svm hypothesis is similar to logistic regression,

You can see below the cost function and hypothesis for logistic regression, the the regularisation term $(\lambda/2m) \sum (\theta_j^2)$ is there to make sure we get smaller values of theta

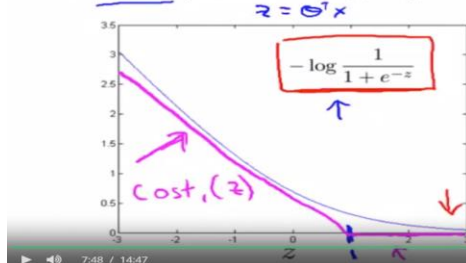
In svm we just choose different hypothesis, these hypothesis are approximations of hypothesis of logistic regression for case $y=1$ and 0 , also in cost function we drop m from denominator (svm people just prefer dropping it), also instead of having λ in front of regularisation term we have C in front of hypothesis minimising term, this C act as $1/\lambda$, so giving same effect as regularisation.

Alternative view of logistic regression ^(x,y)

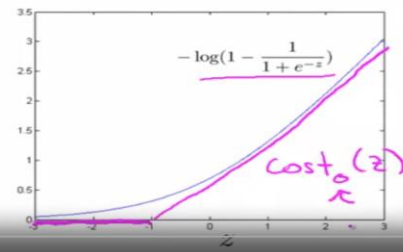
Cost of example: $-(y \log h_{\theta}(x) + (1 - y) \log(1 - h_{\theta}(x))) \leftarrow$

$$= -y \log \frac{1}{1 + e^{-\theta^T x}} - (1 - y) \log \left(1 - \frac{1}{1 + e^{-\theta^T x}}\right) \leftarrow$$

If $y = 1$ (want $\theta^T x \gg 0$):



If $y = 0$ (want $\theta^T x \ll 0$):



Support vector machine

Logistic regression:

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \underbrace{\left(-\log h_{\theta}(x^{(i)})\right)}_{\text{cost}_1(\theta^T x^{(i)})} + (1 - y^{(i)}) \underbrace{\left(-\log(1 - h_{\theta}(x^{(i)}))\right)}_{\text{cost}_0(\theta^T x^{(i)})} \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Support vector machine:

$$\min_{\theta} \sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) + \frac{\lambda}{2} \sum_{j=1}^n \theta_j^2$$

$\min_u (u-5)^2 + 1 \rightarrow u=5$
 $\min_u 10(u-5)^2 + 10 \rightarrow u=5$

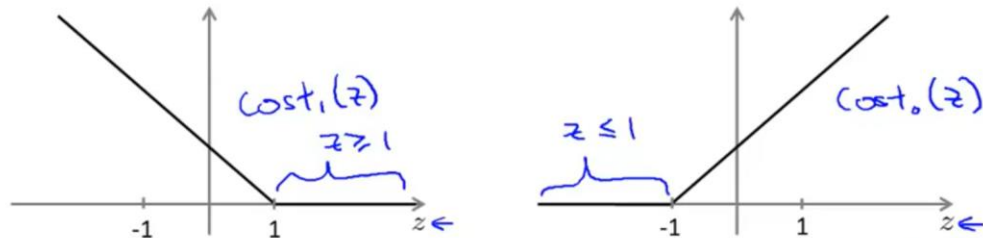
$A + \lambda B \leftarrow$
 $C = \frac{1}{\lambda}$
 $\rightarrow C A + B \leftarrow$

More Intuition on SVM

So you can think of a case in SVM when C is very large, so in this case what will happen is that in order to minimise J , the first term of cost function has to go to zero, and that hypothesis term goes to zero if we have $\theta^T x < -1$ if case of $y=0$ and $\theta^T x > 1$ is case of $y=1$, so

Support Vector Machine

$$\rightarrow \min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \underline{\text{cost}_1(\theta^T x^{(i)})} + (1 - y^{(i)}) \underline{\text{cost}_0(\theta^T x^{(i)})} \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$



\rightarrow If $y = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0)

$$\theta^T x \geq 1$$

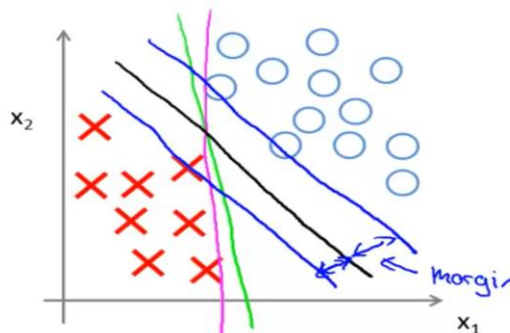
\rightarrow If $y = 0$, we want $\theta^T x \leq -1$ (not just < 0)

$$\theta^T x \leq -1$$

consider a case

The Svm algo due to its nature of choosing $\theta^T x > 1$, where $\theta^T x > 0$ also would have worked, makes sure that we get a very distinctive fit, so this means we will get more separation or margin between 0 and 1 output, **thus svm is called large margin classifier**

SVM Decision Boundary: Linearly separable case



Large margin classifier

this large margin classifier. But

Andrew Ng

SVM intuition via geometry

If you look closely for cases when C is very large we get to a cost function that is minimisation of $(\frac{1}{2}) * \sum(\theta_j^2)$, keeping this mind we also see that,

$\theta^T x$ is nothing but dot product of θ vector and x vector, and if look closely then,

$\theta^T x = p * \text{mod}(\theta)$, where p is projection of x vector on θ vector which is as you have studied in jee is $p = x \cdot \theta(\text{unit vector}) = x \cdot \theta / \text{mod}(\theta)$

So now you can see the power of this projection p , now we know that $\theta^T x > 1$ for $y=1$, so that means $p \cdot \|\theta\| > 1$, but suppose if p is small then this would mean that θ has to be large for inequality to satisfy, but if θ large would mean increase of cost function, so this wouldn't happen, thus svm will try to keep p big, and making this p big ensures large margin between yes and no boundaries

So magnitude of p is like margin, greater of it ensures more margin, thus lesser θ which is what we want to minimise cost function in case when $C \gg 1$

SVM Decision Boundary

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} (\theta_1^2 + \theta_2^2) = \frac{1}{2} (\sqrt{\theta_1^2 + \theta_2^2})^2 = \frac{1}{2} \|\theta\|^2$$

s.t. $\theta^T x^{(i)} \geq 1$ if $y^{(i)} = 1$
 $\rightarrow \theta^T x^{(i)} \leq -1$ if $y^{(i)} = 0$

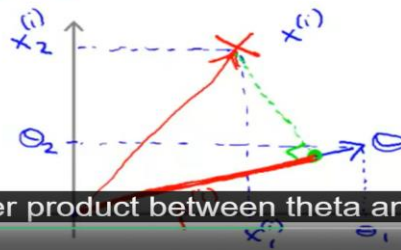
Simplification: $\theta_0 = 0$. $n=2$

$\omega = (\sqrt{\omega})^2$

$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \theta_0 = 0$

$$\theta^T x^{(i)} = ?$$

$\uparrow \quad \uparrow$
 $u^T v$



$$\theta^T x^{(i)} = p^{(i)} \cdot \|\theta\| \leftarrow$$

$$= \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} \leftarrow$$

inner product between theta and $X(i)$.

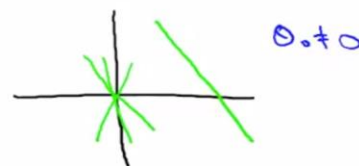
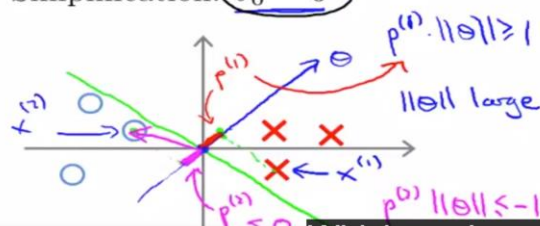
SVM Decision Boundary

$$\rightarrow \min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} \|\theta\|^2 \leftarrow$$

$$\text{s.t. } \begin{cases} p^{(i)} \cdot \|\theta\| \geq 1 & \text{if } y^{(i)} = 1 \\ p^{(i)} \cdot \|\theta\| \leq -1 & \text{if } y^{(i)} = 0 \end{cases}$$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ .

Simplification: $\theta_0 = 0$



$$p^{(i)} \cdot \|\theta\| \geq 0$$

$\rightarrow \|\theta\|$ can be smaller.



Which again corresponds to the

SVM hypothesis

Kernels

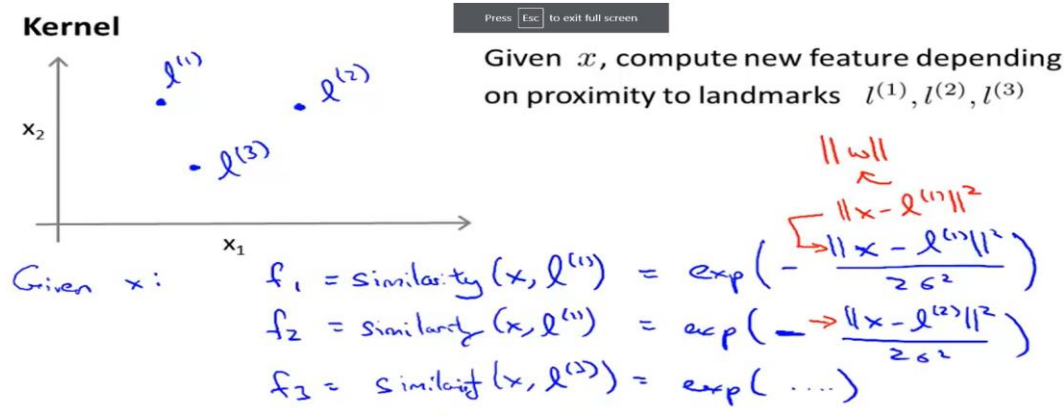
There was a question in logistic regression that if you have a non-linear decision boundary then you have to increase your feature like cross features $x_1 * x_2$ in order to fit the data, but as we realise in Computer vision that cross feature will substantially increase the numbers of features, so is there are features that are better than cross features and features that help classification a great bit, it turns out there is a mathematical way to define such features

We will call these features given by similarity function, in this case we will use similarity function that are called gaussian kernels, kernels is synonymous to similarity function,

So we pick some samples from input set, i.e. pick vector x , call them l_1, l_2, l_3 , l_1 is set of feature, l_2 is also set of feature for different input set.

Now we define gaussian kernel as $f_1 = \exp(-(x-l)^2/2*\sigma^2)$, so f_1 is our new feature similarly, f_2 and f_3 ,

What that similarity function gives us is that, if x is close to l , then $f_1 = 1$, otherwise if x is far away from l then $f_1 = 0$

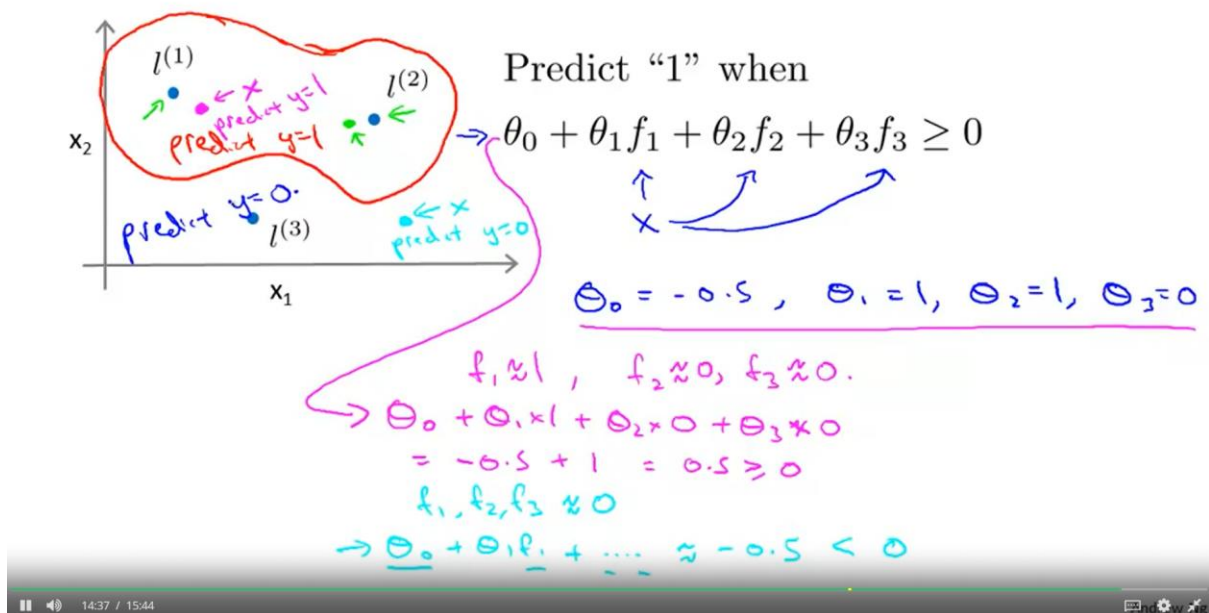


Andrew Ng

So we define decision boundary by $\theta_0 + f_1 * \theta_1 + f_2 * \theta_2 \dots$

So if our new x point is far away from all l 's, so we get f_1, f_2, f_3 all zero and what we are left with is just θ_0 , which is -ve thus $\theta * f < 0$, thus $y=0$

Also if say a point is close to l_1 so all other $\theta * f$ will be zero but, $\theta_1 * f_1$ will be 1, so $-0.5 + 1$ will be positive so, $y=1$, thus this way we are able to predict $y=1$ for points close to l 's and $y=0$ otherwise



SVM cost function with kernels

One way to pick l 's or landmark is choose every x row as your landmark, so you will get n l 's thus you will get n f features

$x_{11}, x_{12}, x_{13}, y_1$

$x_{21}, x_{22}, x_{23}, y_2$

$x_{31}, x_{32}, x_{33}, y_3$, old training set looks like this with x_1, x_2, x_3 as features for each row

$F_1 = f(x_{11}-l_1) + f(x_{12}-l_1) + f(x_{13}-l_1), y_1$

$F_2 = f(x_{21}-l_1) + f(x_{22}-l_1) + f(x_{23}-l_1), y_2$

$F_3 = f(x_{31}-l_1) + f(x_{32}-l_1) + f(x_{33}-l_1), y_3$, so you have new training set like this, where x_1, x_2 feature will be replaced by above

So your hypothesis becomes $y = \text{cost}_1(\theta^T f) + (1-y) \text{cost}_0(\theta^T f)$, so all the rows in dataset are replaced by single f_i feature

SVM with Kernels

- Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$,
 → choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$.

Given example x :

$$\begin{aligned} \rightarrow f_1 &= \text{similarity}(x, l^{(1)}) \\ \rightarrow f_2 &= \text{similarity}(x, l^{(2)}) \\ &\vdots \end{aligned}$$

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \quad f_0 = 1$$

For training example $(x^{(i)}, y^{(i)})$:

$$\begin{aligned} f_1^{(i)} &= \text{sim}(x^{(i)}, l^{(1)}) \\ f_2^{(i)} &= \text{sim}(x^{(i)}, l^{(2)}) \\ &\vdots \\ f_i^{(i)} &= \text{sim}(x^{(i)}, l^{(i)}) = \exp(-\frac{0}{2\sigma^2}) = 1 \end{aligned}$$

$$x^{(i)} \in \mathbb{R}^{n+1} \text{ (or } \mathbb{R}^n) \rightarrow f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \quad f_0^{(i)} = 1$$

So given these kernels

Sometimes svm people just choose to write regularisation term as $\theta_j^2 = \theta_t^T M \theta$, where M is used to control somethings, that author didn't describe much, but author told not to worry abt M ,

SVM with Kernels

Hypothesis: Given x , compute features $f \in \mathbb{R}^{m+1}$ $\theta \in \mathbb{R}^{m+1}$

→ Predict "y=1" if $\theta^T f \geq 0$ $\theta_0 f_0 + \theta_1 f_1 + \dots + \theta_m f_m$

Training:

$$\rightarrow \min_{\theta} C \sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^m \theta_j^2$$

$$\begin{aligned} - \sum_j \theta_j^2 &= \theta^T \theta \leftarrow \theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_m \end{bmatrix} \quad (\text{ignore } \theta_0) \\ &\rightarrow \theta^T M \theta \quad M = 10,000 \end{aligned}$$

Also author told that off the shelf softwares are there to minimise above cost function

Final implementation of SVM

Solve cost function of svm by already available libraries

DO perform feature scaling when finding f_1 , f_2 and so on

Kernel (similarity) functions:

```
function f = kernel(x1, x2)
    f = exp( - (|| x1 - x2 ||^2) / (2 * sigma^2) )
    return
```

Handwritten notes: $x^{(i)}$, $x^{(j)} = x^{(i)}$, $x \rightarrow \begin{matrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{matrix}$

→ Note: Do perform feature scaling before using the Gaussian kernel.

Handwritten notes: $x \in \mathbb{R}^n$

$$\begin{aligned} \Rightarrow \|x - l\|^2 &= \|v\|^2 = v_1^2 + v_2^2 + \dots + v_n^2 \\ &= (x_1 - l_1)^2 + (x_2 - l_2)^2 + \dots + (x_n - l_n)^2 \end{aligned}$$

Handwritten notes: $\underbrace{1000 \text{ sq. ft}}_{1-5 \text{ bedrooms}}$

This could be like a thousand squared,

Logistic regression vs. SVMs

Press Esc to exit full screen

n = number of features ($x \in \mathbb{R}^{n+1}$), m = number of training examples

→ If n is large (relative to m): (e.g. $n \geq m$, $n = 10,000$, $m = 10 \dots 1000$)

→ Use logistic regression, or SVM without a kernel ("linear kernel")

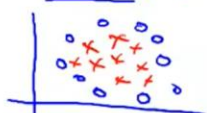
→ If n is small, m is intermediate: ($n = 1-1000$, $m = 10-10,000$) ←

→ Use SVM with Gaussian kernel

If n is small, m is large: ($n = 1-1000$, $m = 50,000+$)

→ Create/add more features, then use logistic regression or SVM without a kernel ↑

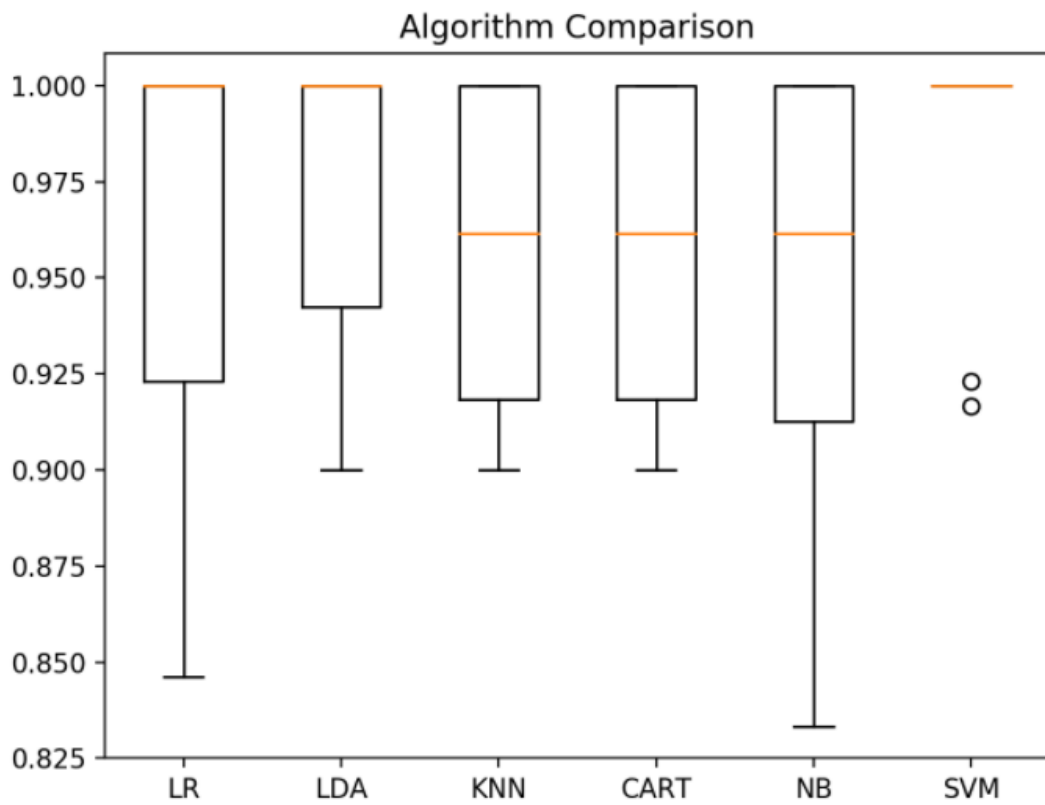
→ Neural network likely to work well for most of these settings, but may be slower to train.



The one disadvantage, or the one

Section 6 – Algorithm Comparison

Winner – Support Vector Machine Algorithm, as it showed greatest accuracy



Note : not the final documentation of code, will refine on it.

