VI. Logistic Regression

Lecture 33: Classification

We will start with a binary classification problem:

→ Email: Spam/Not Spam?

→ Online transactions: Fraudulent (Yes/No)?

→ Tumor: Malignant/Benign?

We can represent the output as

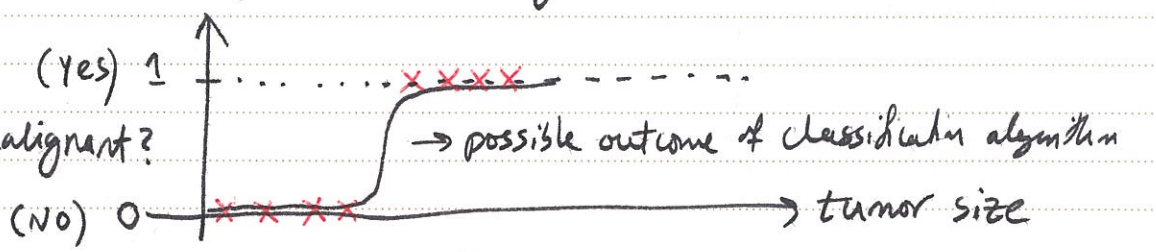
$$y \in \{0, 1\}$$

"Negative class" e.g. benign tumor

"positive class" e.g. malignant tumor.

→ Later on we will generalize to multi-classification problems where e.g. $y \in \{0, 1, 2, 3\}$

Here's an example of a training set:



* Any curve that would be a good fit to these points ought to be non-linear. Of course, a non-linearity of this sort is very hard to capture with a polynomial, as it would require many terms. It is fairly clear, therefore, that linear regression is not going to do.

* It would make sense that in a classification problem we predict probabilities. e.g. what is the probability that a 2mm tumor is malignant?

Then in this light, the fitting curve drawn in the previous page can be interpreted as the the probability, as a function of tumor size, that the tumor is malignant. Choosing $y = \{0, 1\}$ to correspond to {benign, malignant} also makes more sense now.

* If we have a probabilistic interpretation, we would want our hypothesis function to be always between 0 & 1.

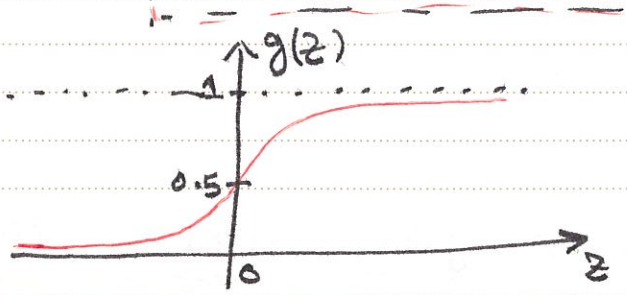
Lecture 34: Hypothesis Representation

→ Logistic Regression is a type of classification problem.

→ The hypothesis function used in logistic regression is:

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

↑ Sigmoid function
Logistic function



* Note that $0 \leq h_\theta(x) \leq 1$ as desired.

* As we mentioned before: $h_\theta(x)$ = estimated probability that $y=1$ on input x.

$$h_\theta(x) = P(y=1 \mid x; \theta)$$

Probability of $y=1$ given x & parametrized by θ.

* Obviously $P(y=0 \mid x; \theta) = 1 - P(y=1 \mid x; \theta)$

$$= 1 - h_\theta(x)$$

Lecture 35: Decision Boundary

2 → Although we're interested in a probabilistic prediction in classification problems, often we'd want to make concrete decisions based on the predicted probability once we're encountered with some new input. The simplest thing to do is:

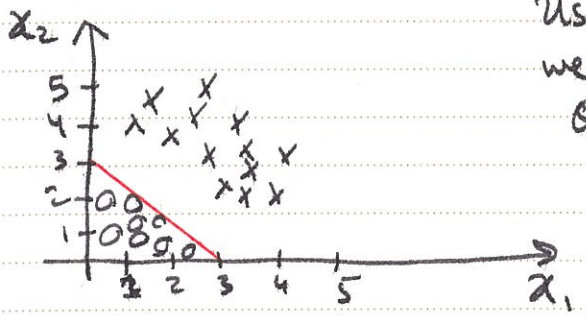
$$\text{Decide} \begin{cases} y=1 & \text{if } P(y=1 \mid x; \theta) \geq 0.5 \\ y=0 & \text{if } P(y= \mid x; \theta) < 0.5 \end{cases}$$

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

Now since $h_\theta(x) = \dfrac{1}{1+e^{-\theta^T x}}$: $\begin{cases} h_\theta(x) \geq 0.5 \leftrightarrow \theta^T x \geq 0 \\ h_\theta(x) < 0.5 \leftrightarrow \theta^T x < 0 \end{cases}$

∴ The decision boundary is then defined via $\theta^T x = 0$ *

Let's say we have two feature x_1 & x_2 & our training set is as follows:



Using $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$, we (somehow) fit to the data & find $\theta_0 = -3$, $\theta_1 = \theta_2 = 1$.

Our decision boundary is then

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$$

$$\leftrightarrow \quad x_1 + x_2 = 3$$

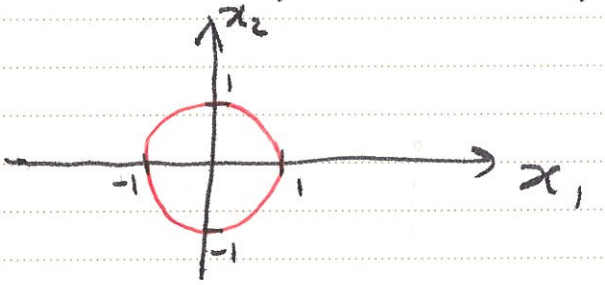
$$\begin{cases} y = 1 : & -3 + x_1 + x_2 \geq 0 \\ y = 0 : & -3 + x_1 + x_2 \leq 0 \end{cases}$$

→ Note that the decision boundary is a property of the hypothesis function & the values of θ, which are of course determined by fitting the model to the training set.

→ With more complicated models, one can have more complicated decision boundaries:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

& $\theta_0 = -1$; $\theta_1 = \theta_2 = 0$; $\theta_3 = \theta_4 = 1$



Lecture 36: Cost Function ⊕ Lecture 37: Simplified Cost function and Gradient Descent

→ How do we fit the parameters θ in logistic regression?

→ Remember that $h_\theta(x)$ is interpreted as the probability that $y=1$ at x. What is the probability of obtaining the training set $\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\dots,(x^{(m)},y^{(m)})\}$?

$$x^{(i)}=\begin{bmatrix}1\\ x_1^{(i)}\\ \vdots\\ x_n^{(i)}\end{bmatrix} \quad \& \quad y^{(i)}\in\{0,1\}$$

The probability of $(x^{(i)},y^{(i)})$ happening is:

$$P^{(i)}=\begin{cases} h_\theta(x^{(i)}) & \text{if } y^{(i)}=1\\ 1-h_\theta(x^{(i)}) & \text{if } y^{(i)}=0\end{cases}$$

or more condensely $P^{(i)}=h_\theta(x^{(i)})^{y^{(i)}}\left(1-h_\theta(x^{(i)})\right)^{1-y^{(i)}}$

→ The probability of obtaining the whole training set is then:

$$P(\theta)=\prod_{i=1}^{m}P^{(i)}=\prod_{i=1}^{m}\left[h_\theta(x^{(i)})^{y^{(i)}}\left(1-h_\theta(x^{(i)})\right)^{1-y^{(i)}}\right]$$

* we can now use the Principle of maximum likelihood to find the value of θ which maximizes P. *

Note that if we let $\tilde{P}(\theta)\equiv-\ln P(\theta)$

→ $\dfrac{\partial\tilde{P}}{\partial\theta_k}=-\dfrac{1}{P(\theta)}\dfrac{\partial P}{\partial\theta_k}\Rightarrow\dfrac{\partial P}{\partial\theta_k}=0\leftrightarrow\dfrac{\partial\tilde{P}}{\partial\theta_k}=0$ since $P(\theta)>0$

Therefore, all extrema of P & $\tilde{P}$ are the same. Moreover, if θ_{max} maximizes P, it minimizes $\tilde{P}$. It is more convinient to minimize $\tilde{P}$, & this is what we'll do:

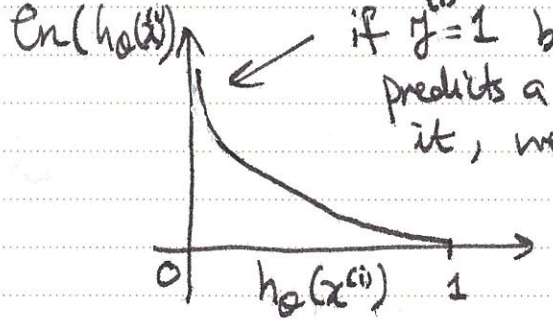
→ The cost function for logistic regression is defined as:

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

$$\text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right) \equiv -y^{(i)} \ln\left(h_\theta(x^{(i)})\right) - (1-y^{(i)}) \ln\left(1-h_\theta(x^{(i)})\right)$$

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

is the cost we charge the algorithm for training set i.



if $y^{(i)}=1$ but the model predicts a very low probability for it, we charge a large cost.

→ Let's show that for $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$, $J(\theta)$ is convex.

$$\frac{\partial h}{\partial \theta_k} = \frac{e^{-\theta^T x}}{(1+e^{-\theta^T x})^2} x_k = e^{-\theta^T x} h_\theta^2(x) x_k$$

$$\frac{\partial J}{\partial \theta_k} = \frac{-1}{m} \sum_{i=1}^{m} \frac{y^{(i)}}{h_\theta(x^{(i)})} \frac{\partial h_\theta(x^{(i)})}{\partial \theta_k} - \frac{(1-y^{(i)})}{1-h_\theta(x^{(i)})} \frac{\partial h_\theta}{\partial \theta_k}(x^{(i)})$$

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

$$\frac{\partial J}{\partial \theta_k} = \frac{-1}{m} \sum_{i=1}^{m} \frac{\partial h_\theta(x^{(i)})}{\partial \theta_k} \left\{ \frac{y^{(i)}}{h_\theta(x^{(i)})} - \frac{(1-y^{(i)}) e^{\theta^T x}}{h_\theta(x^{(i)})} \right\}$$

$$= \frac{-1}{m} \sum_{i=1}^{m} e^{-\theta^T x^{(i)}} h_\theta^2(x^{(i)}) x_k^{(i)} \cdot \frac{y^{(i)}[1+e^{\theta^T x}] - e^{\theta^T x}}{h_\theta(x^{(i)})}$$

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

$$\boxed{\frac{\partial J}{\partial \theta_k} = \frac{1}{m} \sum_{i=1}^{m} [h_\theta(x^{(i)}) - y^{(i)}] x_k^{(i)}}$$

$$\frac{\partial^2 J}{\partial \theta_k \partial \theta_\ell} = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial h_\theta(x^{(i)})}{\partial \theta_\ell} x_k^{(i)}$$

$$= \frac{1}{m} \sum_{i=1}^{m} e^{-\theta^T x^{(i)}} h_\theta^2(x^{(i)}) x_\ell^{(i)} x_k^{(i)}$$

$$\rightarrow \nabla^2 J = \frac{1}{m} \sum_{i=1}^{m} e^{-\theta^T x^{(i)}} h_\theta^2(x^{(i)}) x^{(i)T} x^{(i)} > 0 \checkmark$$

2) $J(\theta)$ only has one minima, i.e the global minima !

Gradient Descent: repeat until convergence {

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

}

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

As with linear regression, feature scaling is important!

Lecture 38: Advanced Optimization

Gradient descent is not the only algorithm we can use to minimize our cost function $J(\theta)$. Here are a few more advanced ones:

- → Conjugate Gradient
- → BFGS
- → L-BFGS

Some advantages of these more advanced algorithms:

- → No need to manually pick the learning rate α.
- → Often faster than Gradient Descent.

There are implementations of these in various languages. They normally take $J(\theta)$ & $\frac{\partial J}{\partial \theta_j}$ as inputs.

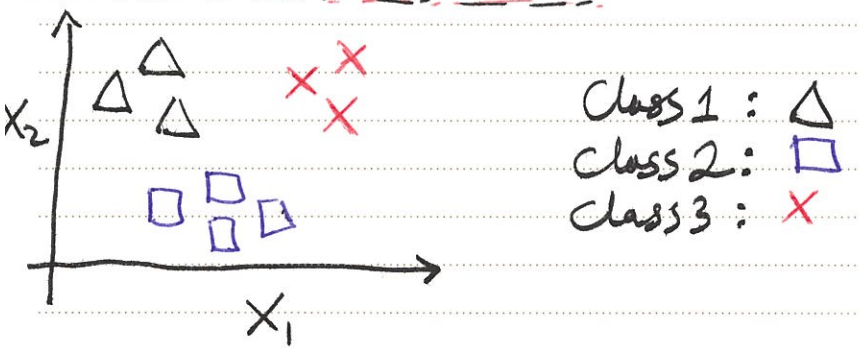
In Octave, 'fminunc' can be used.

Lecture 39: Multiclass Classification: One-vs-all

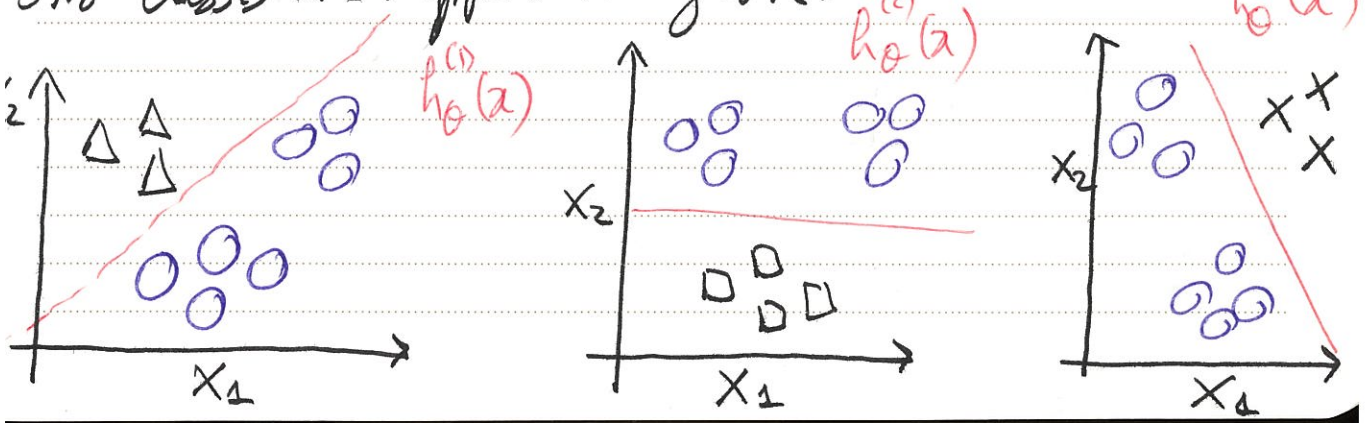
So far we've been discussing binary classification problems.
What if we have multiple classes?

Examples: (i) Email folding/tagging: Work, Friends, Family, Hobby
(ii) Weather prediction: Sunny, Cloudy, Rain, Snow

One-vs-all (one-vs-rest):



For each class, we solve a binary classification problem where the chosen class is taken as positive, and all the classes are mapped to negative:



On a new input x, to make a prediction, pick the class i that has the maximum $h_\theta^{(i)}(x)$:

$$\max_i h_\theta^{(i)}(x)$$

Note that nothing ensures that $\sum_{i=1}^{K} h_\theta^{(i)}(x) = 1$, so in that we should not really think about $h_\theta^{(i)}(x)$ as the probability that class $= i$ at x.

I suspect, though, if these the training set is large, the sum of probabilities will be very close to 1. Why?

Assume we're using logistic regression as our binary classifier

Then we would need to solve: (see page 34)

$$\frac{\partial J^{(1)}}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta^{(1)}(x^{(i)}) - \delta_{y^{(i)},1} \right] x_j^{(i)} = 0$$

$$\vdots$$

$$\frac{\partial J^{(K)}}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta^{(K)}(x^{(i)}) - \delta_{y^{(i)},K} \right] x_j^{(i)} = 0$$

assuming we have K classes: $y^{(i)} \in \{1, 2, \dots, K\} \; \forall i$

Summing up these equations & noting that $\sum_{q=1}^{K} \delta_{y^{(i)}, q} = 1$

$$\sum_{i=1}^{m} \left\{ \sum_{l=1}^{K} h_\theta^{(q)}(x^{(i)}) - 1 \right\} x_j^{(i)} = 0$$

If there are many training examples, i.e. m is very large, it's likely that solving the equation above would require

$$\sum_{q=1}^{K} h_\theta^{(q)}(x) = 1.$$

Again, though, there's no guarantee at all.

There's a multiclass classification problem which does predict probabilities that sum to 1, by construction.
This is not covered in the course but I'll look at it briefly below.

Multinomial Logistic Regression

This is also known as the Maximum Entropy Classifier, Softmax regression, and a bunch of other names.

Suppose: $y \in \{1, 2, 3, \ldots, K-1, K\}$.

Given a training set $\{(x^{(i)}, y^{(i)})\}$, our goal is to predict the following probabilities for an unseen value of x:

probability of $y^{(i)}=1$ → $P_1(x)$, probability of $y^{(i)}=2$ → $P_2(x)$, ..., probability of $y^{(i)}=K$ → $P_K(x)$

Of course, we require that

$$\sum_{q=1}^{K} P_q(x) = 1 \qquad \forall x. \qquad \oplus$$

Multinomial logistic regression uses the following parametrization:

$$P_q(x) = \frac{e^{-\theta_q^T x}}{1 + \sum_{q'=1}^{K-1} e^{-\theta_{q'}^T x}} \qquad q \in [1, K-1]$$

P_K is fixed by $\oplus$ above:

$$P_K(x) = \frac{1}{1 + \sum_{q'=1}^{K-1} e^{-\theta_{q'}^T x}}$$

Our job is now to estimate $\{\theta_q\}$ $q \in [1, K-1]$

Note that every θ_ℓ is an $n+1$ dimensional vector.
(n = # of features.)

We will use the principle of maximum likelihood:
What is the probability of $(x^{(i)}, y^{(i)})$ occuring?

$$p^{(i)} = P_1(x^{(i)})^{\delta_{y^{(i)},1}} P_2(x^{(i)})^{\delta_{y^{(i)},2}} \cdots P_K(x^{(i)})^{\delta_{y^{(i)},K}}$$

$$J(\theta) \equiv -\frac{1}{m} \ln \prod_{i=1}^{m} p^{(i)} = -\frac{1}{m} \sum_{i=1}^{m} \ln p^{(i)}$$

$$= -\frac{1}{m} \sum_{i=1}^{m} \left[\delta_{y^{(i)},1} \ln P_1(x^{(i)}) + \cdots + \delta_{y^{(i)},K} \ln P_K(x^{(i)}) \right]$$

$$= \frac{-1}{m} \sum_{i=1}^{m} \left[\delta_{y^{(i)},1} \ln \frac{e^{-\theta_1^T x^{(i)}}}{1 + \sum_{\ell=1}^{K-1} e^{-\theta_\ell^T x^{(i)}}} \right.$$

$$+ \delta_{y^{(i)},2} \ln \frac{e^{-\theta_2^T x^{(i)}}}{1 + \sum_{\ell=1}^{K-1} e^{-\theta_\ell^T x^{(i)}}}$$

$$+ \cdots$$

$$+ \delta_{y^{(i)},K-1} \ln \frac{e^{-\theta_{K-1}^T x^{(i)}}}{1 + \sum_{\ell=1}^{K-1} e^{-\theta_\ell^T x^{(i)}}}$$

$$\left. + \delta_{y^{(i)},K} \ln \frac{1}{1 + \sum_{\ell=1}^{K-1} e^{-\theta_\ell^T x^{(i)}}} \right]$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\Big\{\delta_{y^{(i)},1}\Big[-\theta_1^T x^{(i)} - \ln\Big[1+\sum_{\ell=1}^{K-1} e^{-\theta_\ell^T x^{(i)}}\Big]\Big]$$

$$+ \cdots$$

$$+ \delta_{y^{(i)},K-1}\Big[-\theta_{K-1}^T x^{(i)} - \ln\Big[1+\sum_{\ell=1}^{K-1} e^{-\theta_\ell^T x^{(i)}}\Big]\Big]$$

$$+ \delta_{y^{(i)},K}\, x - \ln\Big[1+\sum_{\ell=1}^{K-1} e^{-\theta_\ell^T x^{(i)}}\Big]\Big\}$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\Big[\sum_{\ell=1}^{K-1}\delta_{y^{(i)},\ell}\,\theta_\ell^T x^{(i)} + \ln\Big[1+\sum_{\ell=1}^{K-1} e^{-\theta_\ell^T x^{(i)}}\Big]\Big]$$

where we have used the fact that $\sum_{\ell=1}^{K}\delta_{y^{(i)},\ell} = 1$

We can think about $\{\theta_\ell\}$ as a $(n+1)\times(K-1)$ matrix:

$$\textcircled{H}_{j\ell} = [\theta_\ell]_j \qquad \textcircled{H} = \big[[\theta_1][\theta_2]\cdots[\theta_{K-1}]\big]$$

rows are features ← → classes

$$\theta_\ell^T x^{(i)} = \sum_{j=1}^{n+1}[\theta_\ell]_j\, x_j^{(i)} = \sum_{j=1}^{n+1}\textcircled{H}_{j\ell}\, x_j^{(i)}$$

$$\frac{\partial J}{\partial \Theta_{jq}} = \frac{1}{m} \sum_{i=1}^{m} \delta_{y^{(i)},q} x_j^{(i)} + \frac{e^{-\Theta_q^T x^{(i)}}}{1 + \sum_{q=1}^{K-1} e^{-\Theta_q^T x^{(i)}}} x - x_j^{(i)} \Big]$$

$$\boxed{\frac{\partial J}{\partial \Theta_{jq}} = \frac{1}{m} \sum_{i=1}^{m} \left[\delta_{y^{(i)},q} - p_q(x^{(i)}) \right] x_j^{(i)}}$$