Week 3 Lecture Notes

[ML: Logistic Regression:]

Now we are switching from regression problems to classification problems. Don't be confused by the name "Logistic Regression"; It's named this way for historical reasons and is actually an approach to classification problems, not regression problems.

- Binary Classification:

Instead of output vector y being a continuous range of values, it will only be 0 or 1.

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

Where 0 is usually taken as the "negative class" and 1 as the "positive class", but we can do it to assign any representation to it.

We're starting by doing this class for now, called a "Binary Classification Problem."

One method is to use Linear Regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. This method doesn't work well because classification is not actually a linear function.

Hypothesis Representation

Our hypothesis should satisfy:

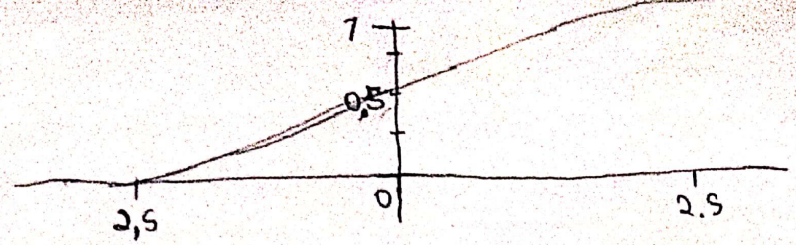
$$0 \leq h_\theta(x) \leq 1$$

Our new form uses the "Sigmoid Function", also called the "Logistic Function":

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

$$z = \theta^T x$$

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



The function $g(z)$, shown here, maps any real number to the $(0,1)$ interval, making it useful for transforming an arbitrary-valued function into a function better suited for classification.

We start with our old hypothesis (linear regression), except that we want to restrict the range to 0 and 1. This is accomplished by plugging $\theta^T x$ into the logistic function.

h_θ will give us the probability that our output is 1. For example $h_\theta(x) = 0.7$ gives us the probability of 70% that our output is 1.

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

$$P(y=0 \mid x;\theta) + P(y=1 \mid x;\theta) = 1$$

The probability that our ~~prediction~~ prediction is 0 is just the complement of our probability that it is 1 (e.g. if probability that it is 1 is 70%, then the probability that it is 0 is 30%).

Decision Boundary:

In order to get our discrete 0 or 1 classification, we can translate the output of the hypothesis function as follows:

$$h_\theta(x) \geq 0.5 \rightarrow y = 1$$

$$h_\theta(x) < 0.5 \rightarrow y = 0$$

The way our logistic function g behaves is that when its input is greater than or equal to zero, its output is greater than or equal to 0.5:

$$g(z) \geq 0.5$$

when $z \geq 0$

...ion Boundary : Video ①

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

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

Suppose predict "$y=1$" if $h_\theta(x) \geq 0.5$

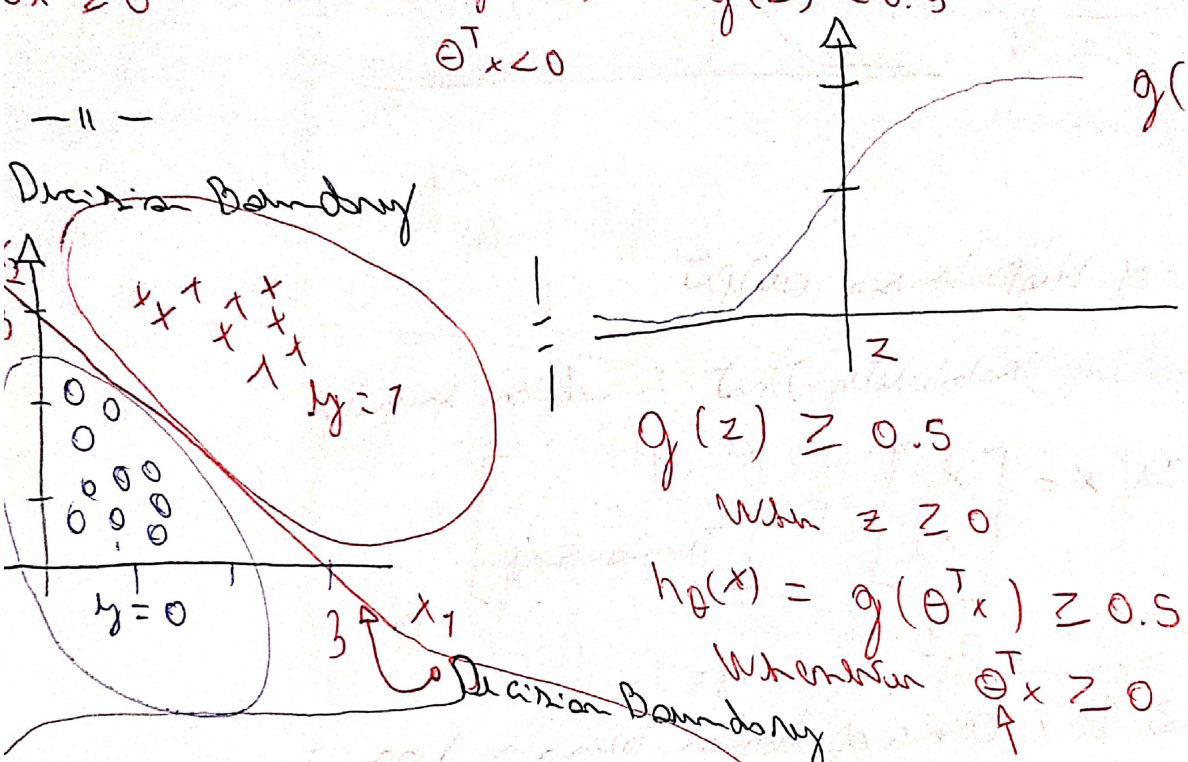
Predict "$y=0$" if $h_\theta(x) < 0.5$

$h_\theta(x) = g(\theta^T x)$ $\quad g(z) < 0.5$

$\theta^T x < 0$

$\theta^T x \geq 0$

— " —



$g(z) \geq 0.5$

When $z \geq 0$

$h_\theta(x) = g(\theta^T x) \geq 0.5$

Whenever $\theta^T x \geq 0$

z

Decision Boundary

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

$\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

Predict "$y=1$" if $\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0$

$x_1 + x_2 \geq 3$

$h_\theta(x) = 0.5$

$x_1 + x_2 = 3$

Straight line.

$x_1 + x_2 < 3$

$y = 0$

Hypothesis Representation : Video

Logistic Regression Model.

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

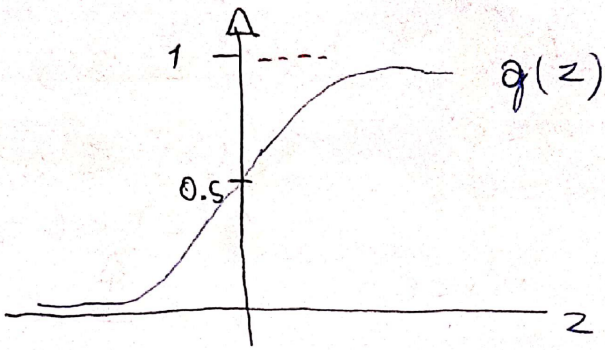
$h_\theta(x) = g(\theta^T x)$

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

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

Sigmoid function

Logistic function



Parameter θ.

Interpretation of Hypothesis output

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

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

$h_\theta(x) = 0.7$

Tell patient that 70% chance of tumor being malignant

$y = 1$

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

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

$y = 0$ or $y = 1$

$P(y=0 \mid x; \theta) + P(y=1 \mid x; \theta) = 1$

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

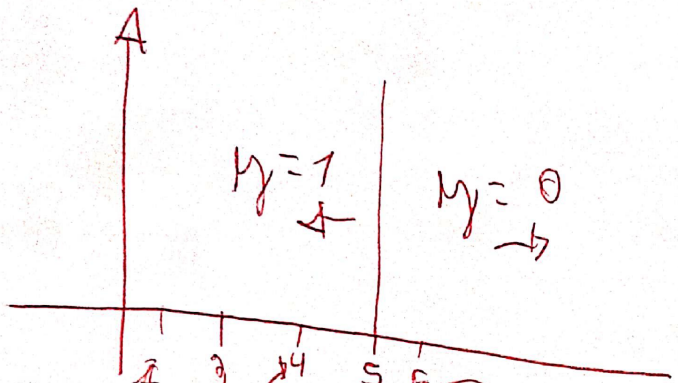
Linear Equation: $g(5 + 1 + 0)$

$\theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$

$h_\theta(x) = g(5 - x_1)$

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

$x_1 + x_2 =$



Concept

Predict $y = 0$ if x_1, x_2 equation than 5.

$g(-6 + 1 + 0)$

$g(-6 + x_1 + x_2)$

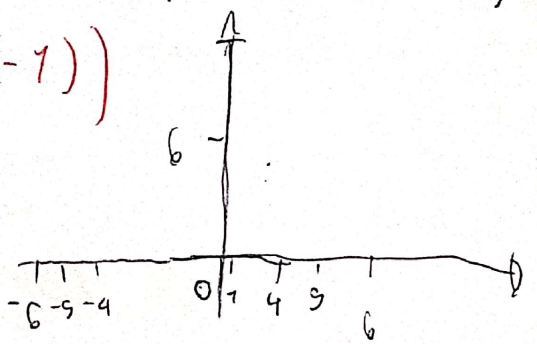
$g(5 - 0)$

$g(5 - (-1))$

$g(5 - 1)$

$g(5 - 2)$

$g(5 - 3)$



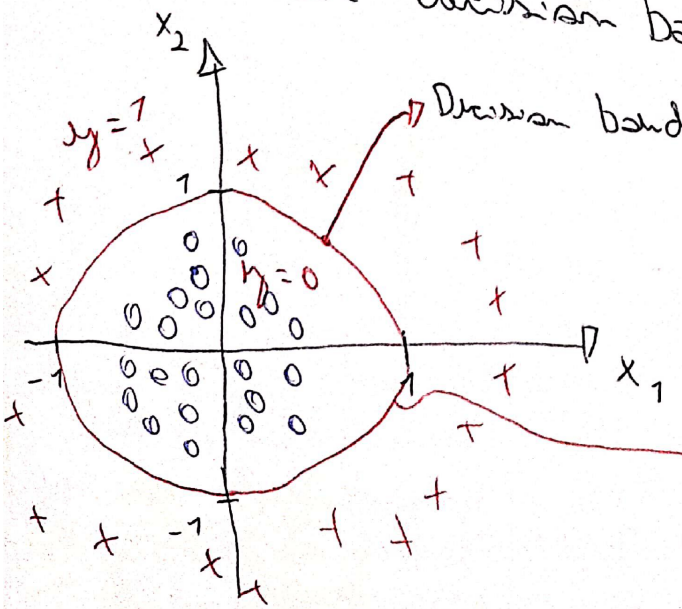
$g(-6+1) = -5$ $g(-6+2) = -4$

$g(-6+10) = 4$ $g(-6+6) = 0$

$g(-6+1) = 1$ $g(-6+0) = -6$

— || —

Non-linear decision boundaries:



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

$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

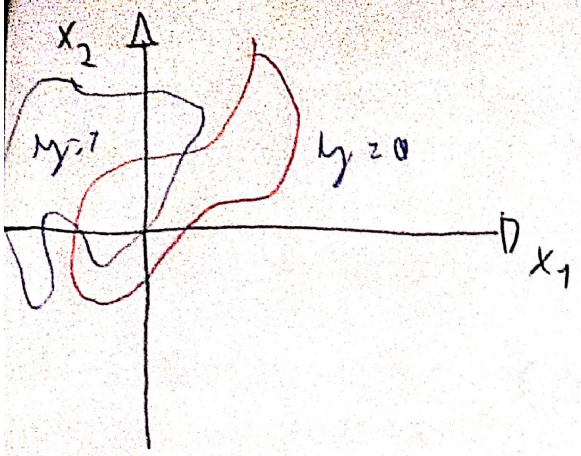
Predict "$y = 1$" if $x_1^2 + x_2^2 = 1$

⇒ Circle

$-1 + x_1^2 + x_2^2 \geq 0$

$x_1^2 + x_2^2 \geq 1$

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



4. Classification:

Email: Spam / Not Spam?

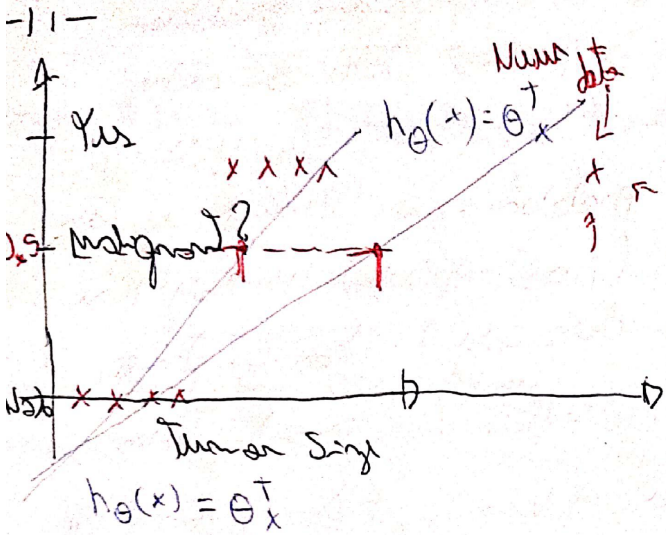
Online Transactions: Fraudulent (Y/N)?

Tumor: Malignant / Benign?

0: "Negative class"

$y \in \{0, 1\}$ 1: "Positive class"

$y \in \{0, 1, 2, 3\}$



Threshold classifier output $h_\theta(x)$ at 0.5:

If $h_\theta(x) \geq 0.5$, predict "$y = 1$"

If $h_\theta(x) < 0.5$, " "$y = 0$"

Classification: $y = 0$ or 1

$h_\theta(x)$ can be > 1 or < 0

Logistic Regression: $0 \leq h_\theta(x) \leq 1$

↳ Classification

Note:

Isn't a good idea to use linear regression in a classification problem

Reading:

Classification:

To attempt classification, one method is to use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. However, this method doesn't work well because classification is not actually a linear function.

The classification problem is just like the regression problem, except that the values we now want to predict take on only a small number of discrete values. For now, we will focus on the binary classification problem in which y can take on only two values, 0 and 1. (Most of what we say here will also generalize to the multiple-class case.) For instance, if we are trying to build a spam classifier for email, then $x^{(i)}$ may be some features of a piece of email, and y may be 1 if it is a piece of spam mail, and 0 otherwise. Hence, $y \in \{0, 1\}$. 0 is also called the negative class, and 1 the positive class, and they are sometimes also denoted by the symbols "−" and "+". Given $x^{(i)}$, the corresponding $y^{(i)}$ is also called the label for the training example.

intuition.

$z = 0, e^0 = 1 \Rightarrow g(z) = 1/2$

$z \to \infty, e^{-\infty} \to 0 \Rightarrow g(z) = 1$

$z \to -\infty, e^{\infty} \to \infty \Rightarrow g(z) = 0$

So if our input to g is $\theta^T x$, then that means:

$h_\theta(x) = g(\theta^T x) \geq 0.5$

When $\theta^T x \geq 0$

From these statements we can make by:

$\theta^T x \geq 0 \Rightarrow y = 1$

$\theta^T x < 0 \Rightarrow y = 0$

The decision boundary is the line that separates the area where $y=0$ and where $y = 1$. It is created by our hypothesis function.

Example:

$$\theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$$

$y = 1$ if $5 + (-1) x_1 + 0 x_2 \geq 0$

$5 - x_1 \geq 0$

$-x_1 \geq -5$

$x_1 \leq 5$

In this case, our decision boundary is a straight vertical line placed on the graph where $x_1 = 5$, and everything to the left of that denotes $y = 1$, while everything to the right denotes $y = 0$.

Again, the input to the sigmoid function $g(z)$ (e.g. $\theta^T x$) doesn't need to be linear, and could be a function that describes a circle (e.g. $z = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$) or any shape to fit our data.

Multi-class Classification: One-vs-all: Video 8

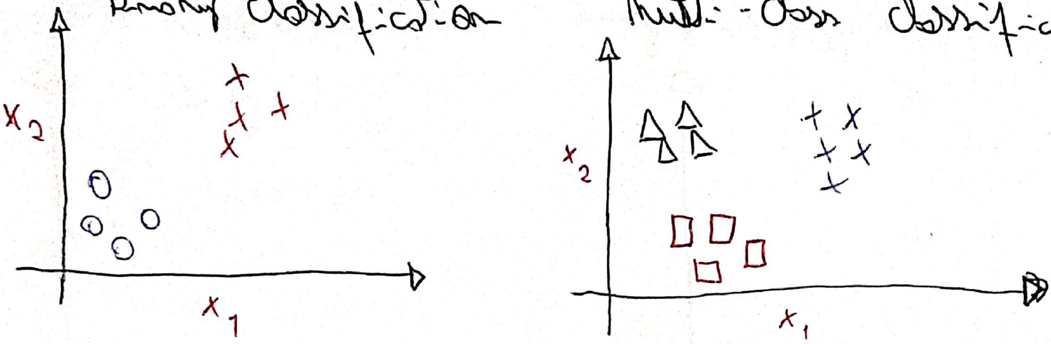
Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby
$y=1$, $y=2$, $y=3$, $y=4$

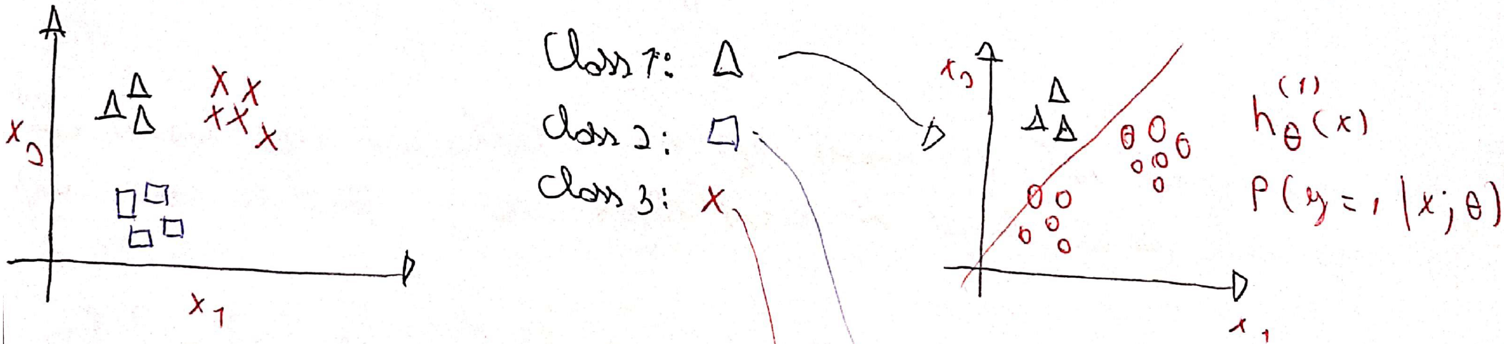
Medical diagnosis: Not ill, Cold, Flu
$y=1$, 2, 3

Weather: Sunny, Cloudy, Rain, Snow
$y=1$, 2, 3, 4

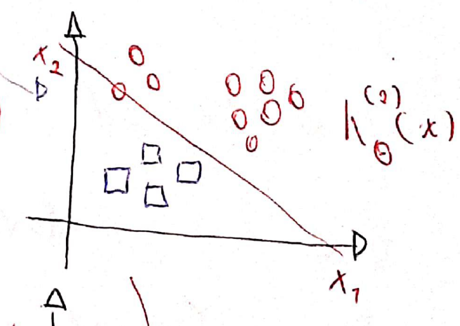
Binary classification — Multi-class classification



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



$$h_\theta^{(i)}(x) = P(y=i|x;\theta) \quad (i=1,2,3)$$



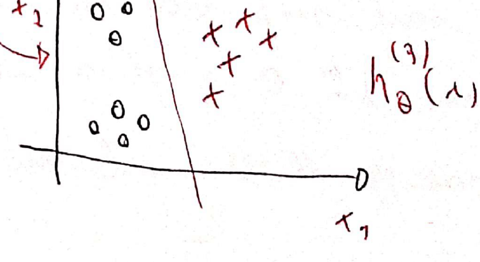
— // —

One-vs-all

Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y=i$.

On a new input x, to make a prediction, pick the class i that maximizes

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



Cost function:

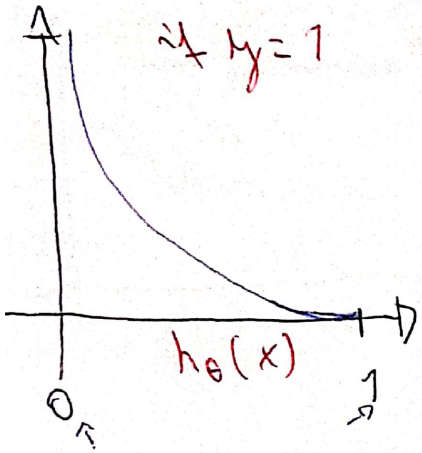
We cannot use the same cost function that we use for linear regression because the Logistic function will cause the output to be wavy, causing many local optima. In other words, it will not be a convex function.

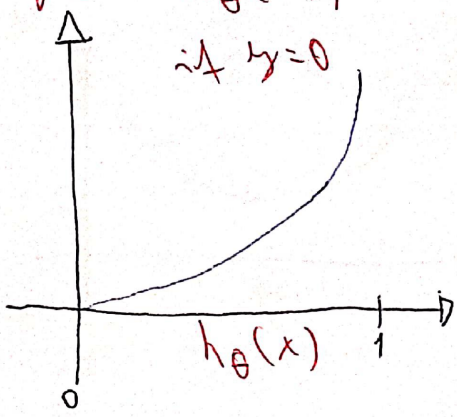
Instead, our cost function for logistic regression looks like:

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

$$\text{Cost}\left(h_\theta(x), y\right) = -\log\left(h_\theta(x)\right) \quad \text{if } y = 1$$

$$\text{Cost}\left(h_\theta(x), y\right) = -\log\left(1 - h_\theta(x)\right) \quad \text{if } y = 0$$





The more our hypothesis is off from y, the larger the cost function output. If our hypothesis is equal to y, then our cost is 0:

$$\text{Cost}\left(h_\theta(x), y\right) = 0 \text{ if } h_\theta(x) = y$$

$$\text{Cost}\left(h_\theta(x), y\right) \to \infty \text{ if } y = 0 \text{ and } h_\theta(x) \to 1$$

$$\text{Cost}\left(h_\theta(x), y\right) \to \infty \text{ if } y = 1 \text{ and } h_\theta(x) \to 0$$

If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity.

Note that writing the cost function in this way guarantees that $J(\theta)$ is convex for logistic regression.

Function Video 8

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

m examples $x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$ $x_0 = 1, y \in \{0, 1\}$

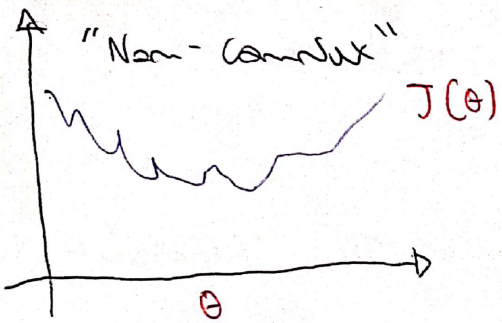
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

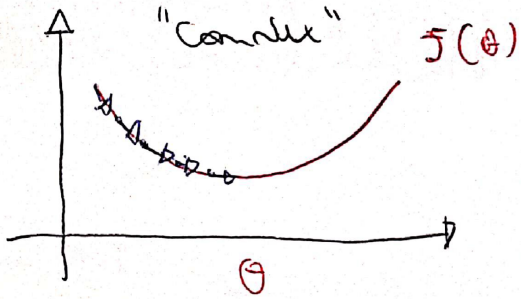
How to choose parameters θ?

-11-

Cost function

~~Linear~~ Logistic regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$





simplified Cost function and Gradient Descent:

We can compress our cost function's two conditional cases into one case:

$$Cost(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

Notice that when y is equal to 1, then the second term $(1-y)\log(1-h_\theta(x))$ will be zero and will not affect the result. If y is equal to 0, then the first term $-y\log(h_\theta(x))$ will be zero and will not affect the result.

We can fully write out our entire cost function as follows:

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

A vectorized implementation is:

$$h = g(X\theta)$$

$$J(\theta) = \frac{1}{m}\cdot\left(-y^T\log(h) - (1-y)^T\log(1-h)\right)$$

Gradient Descent

Remember that the general form of gradient descent is:

Repeat {

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

}

We can work out the derivative part using calculus to get:

Repeat {

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

}

Notice that this algorithm is identical to the one we used in Linear Regression. We still have to simultaneously update all values in Theta.

A Vectorized implementation is:

$$\theta := \theta - \frac{\alpha}{m} X^T \left(g(X\theta) - \vec{y} \right)$$

Partial derivative of $J(\theta)$

First calculate derivative of sigmoid function (it will be useful while finding partial derivative of $J(\theta)$):

$$\sigma(x)' = \left(\frac{1}{1+e^{-x}} \right)' = \frac{-(1+e^{-x})'}{(1+e^{-x})^2} = \frac{-1' - (e^{-x})'}{(1+e^{-x})^2} = \frac{0 - (-x)'(e^{-x})}{(1+e^{-x})^2}$$

$$= \frac{-(-1)(e^{-x})}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \left(\frac{1}{1+e^{-x}} \right) \left(\frac{e^{-x}}{1+e^{-x}} \right) = \sigma(x) \left(\frac{+1 - 1 + e^{-x}}{1+e^{-x}} \right) =$$

$$\sigma(x) \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}} \right) = \sigma(x)(1 - \sigma(x))$$

Now we are ready to find our resulting partial derivative:

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

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

The Vectorized version:

$$\nabla J(\theta) = \frac{1}{m} \cdot X^T \cdot \left(g(X \cdot \theta) - \vec{y} \right)$$

Logistic Regression: Cost function and Gradient Descent: Video

Note:
$y = 0$ or 1 always

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

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

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

To fit parameters θ:

$$\min_\theta J(\theta)$$

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

To make Prediction given new x:

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

$p(y = 1 \mid x; \theta)$

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

$x_j^{(i)}$

Repeat {

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

} (simultaneously update all θ_j)

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

Question

Suppose you

A reasonable way to make sure the learning rate α is set properly and that the gradient descent is running correctly?

Plot $J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$ as a function of number of iterations and make sure $J(\theta)$ is decreasing on every iteration.

Question 2

Vectorized implementation of $\theta := \theta - \alpha \delta$ (δ is a column vector $\delta \in \mathbb{R}^{n+1}$)

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

- ed optimization:

Optimization algorithms:
- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Clever ~~inner~~ loop

Advantages:
- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:
- More complex

-11-

Example:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

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

$$\theta_1 = 5, \theta_2 = 5$$

$$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

Advanced Optimization:

"Conjugate gradient", "BFGS", and "L-BFGS" are more sophisticated, faster ways to optimize θ that can be used instead of gradient descent. A. Ng suggests not to write these more sophisticated algorithms yourself (unless you are an expert in numerical computing) but use the libraries instead, as they're already tested and highly optimized. Octave provides them.

We first need to provide a function that calculates the following two functions for a given input value θ:

$J(\theta)$

$\frac{\partial}{\partial \theta_j} J(\theta)$

We can write a single function that returns both of these:

```
function [jVal, gradient] = costFunction(theta)
  jVal = [... code to compute J(theta) ...];
  gradient = [... code to compute derivative of J(theta) ...];
end
```

Then we can use octave's "fminunc()" optimization algorithm along with the "optimset()" function that creates an object containing the options we want to send to "fminunc()". (Note: The value for MaxIter should be an integer, not a character string - error in video at 07:30)

```
options = optimset('GradObj', 'on', 'MaxIter', 100);
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] = fminunc(@costFunction, ...
initialTheta, options);
```

We give to the function "fminunc()" our cost function, our initial vector of theta values, and the "options" object that we created beforehand.

Multi-class Classification: One-Vs-All

Now we will approach the classification of data into more than two categories. Instead of $y = \{0, 1\}$ we will expand our definition so that $y = \{0, 1 \dots n\}$.

In this case we divide our problem into $n+1$ ($+1$ because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

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

$h_\theta^{(0)}(x) = P(y = 0 | x; \theta)$

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

...

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

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

We are basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the ~~hypothesis~~ hypothesis that returned the highest value as our prediction.

Regularization:

The Problem of Overfitting

Regularization is designed to address the problem of overfitting.

High bias or underfitting is when the form of our hypothesis function h maps poorly to the trend of the data. It usually caused by a function that is too simple or uses too few features. eg. if we take $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ then we are making an initial assumption that a linear model will fit the training data well and will be able to generalize but that may not be true.

At the other extreme, overfitting or high variance is caused by a hypothesis function that fits the available data but does not generalize well to predict new data.

It is usually caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data.

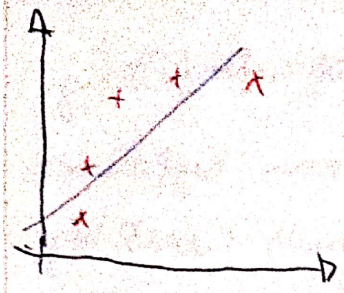
This terminology is applied to both linear and logistic regression.

There are two main options to address this issue of overfitting:

1) Reduce the number of features:
   a) Manually select which features to keep.
   b) Use a model selection algorithm (studied later in the course)
2) Regularization
   keep all features, but reduce the parameters θ_j.
   Regularization works well when we have a lot of slightly useful features.

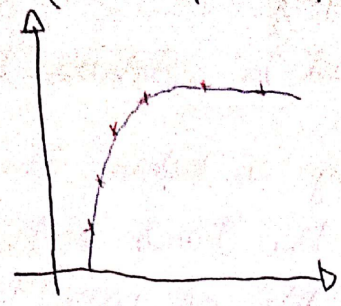
The problem of overfitting: Video:

Example: Linear regression (house prices)



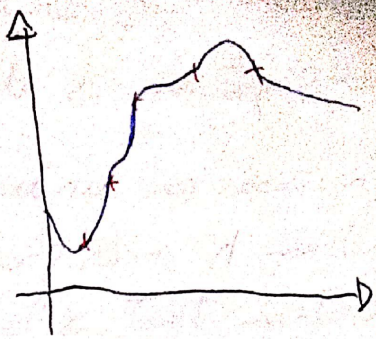
$\rightarrow \theta_0 + \theta_1 x$

"Underfit" "high bias"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"

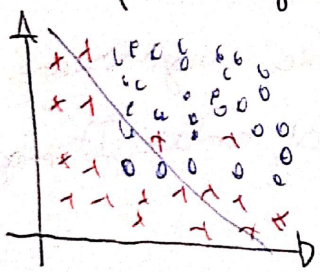


$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"overfit" "high variance"

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \simeq 0$), but fail to generalize to new examples (predict prices on new examples).

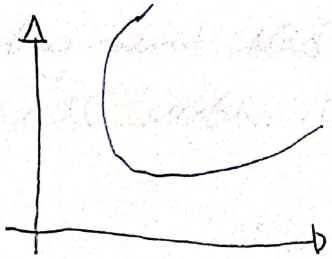
Example: Logistic Regression



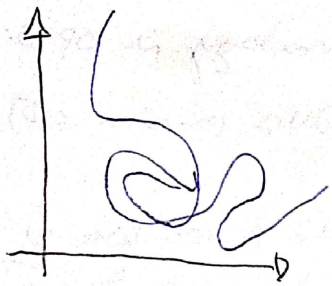
$\rightarrow h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

"Underfit" "high bias"

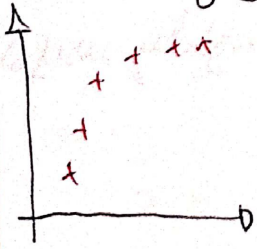


$\rightarrow g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$



$\rightarrow g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

addressing overfitting:



Options to reduce overfit:

- In Next

Features: $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$

Regularization:

If we have overfitting from our hypothesis function, we can reduce the weight that some of the terms in our function convey in creating this cost.

Say we wanted to make the following function more quadratic:

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

We'll want to eliminate the influence of $\theta_3 x^3$ and $\theta_4 x^4$. Without actually getting rid of these features or changing the form of our ~~hypothesis~~ hypothesis, we can instead modify our cost function:

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + 1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2$$

We've added two extra terms at the end to inflate the cost of θ_3 and θ_4. Now, in order for the cost function to get close to zero, we will have to reduce the values of θ_3 and θ_4 to near zero. This will in turn greatly reduce the values of $\theta_3 x^3$ and $\theta_4 x^4$ in our hypothesis function.

We could also regularize all of our theta parameters in a single summation.

$$\min_\theta \frac{1}{2m} \left[\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

The λ, or lambda, is the regularization parameter. It determines how much the costs of our theta parameters are inflated. Using the above cost function with this summation, we can smooth the output of our hypothesis function to reduce overfitting. If lambda is chosen to be too large, it may smooth out the function too much and cause underfitting.

Cost function: Video: [Graphs from the last video]

Suppose we penalize and make θ_3, θ_4 really small.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

$\theta_3 \approx 0 \quad \theta_4 \approx 0$

Will not affect the fitting too much.

Regularization:

Small values for parameters $[\theta_0, \theta_1, \cdots, \theta_n]$

- "Simpler" hypothesis
- Less prone to overfitting

$\theta_3, \theta_4 \approx 0$

Housing:

- Features: $x_1, x_2, \cdots, x_{100}$
- Parameters: $\theta_0, \theta_1, \cdots \theta_{100}$

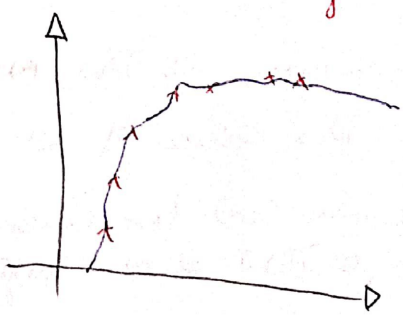
$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \right] + \left[\lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

Regularization Term

good fitting

The data set will $\theta_1, \theta_2, \theta_3 \cdots \theta_{100}$

Regularization Parameter

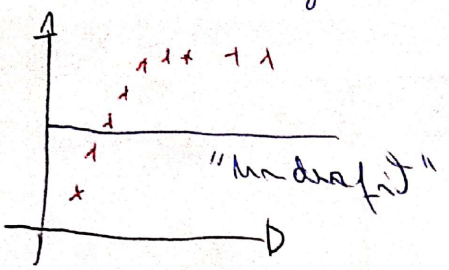


If lambda is too large it will fail to fit the training data. (underfitting)

regularized linear regression, We choose θ to minimize

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?



$\theta_1, \theta_2, \theta_3, \theta_4$

$\theta_1 \simeq 0, \theta_2 \simeq 0$ high bias

$\theta_3 = 0, \theta_4 = 0$

$h_\theta(x)$

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$h_\theta(x) = \theta_0$

Regularized Linear Regression:

We can apply regularization to both linear regression and logistic. We will approach linear regression first.

Gradient Descent:

We will modify our gradient descent function to separate out θ_0 from the rest of the parameters because we do not want to penalize θ_0.

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_0^{(i)}$$

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

↳ $j \in \{1, \dots n\}$

}

The term $\frac{\lambda}{m} \theta_j$ performs our regularization.

With some manipulation our update rule can also be represented as:

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

The first term in the above equation, $1 - \alpha \frac{\lambda}{m}$ will always be less than 1. Intuitively you can see it as reducing the value of θ_j by some amount on every update.

Notice that the second term is now exactly the same as it was before.

Normal Equation:

Now let's approach regularization using the alternate method of the non-iterative normal equation.

To add in regularization, the equation is the same as our original except that we add another term inside the parentheses.

$$\theta = (X^T X + \lambda \cdot L)^{-1} X^T y$$

where $L = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$

- is a matrix with 0 at the top left and 1's down the diagonal, with 0's everywhere else. It should have dimension $(n+1) \times (n+1)$. Intuitively, this is the identity matrix (though we are not including x_0), multiplied with a single number λ.

Recall that if $m < n$, then $X^T X$ is non-invertible. However, when we add the term $\lambda \cdot L$, then $X^T X + \lambda \cdot L$ becomes invertible.

Regularized Linear Regression: Video:

Note: ?

(# examples) × (# features)

Regularized Logistic Regression:

We can regularize logistic regression in a similar way that we regularize linear regression ~~in a similar way~~ let's start with cost function.

Cost function:

Recall that our cost function for logistic regression was:

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

We can regularize this equation by adding a term to the end:

$$+ \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Note well: The second sum, $\sum_{j=1}^{n} \theta_j^2$ means to explicitly exclude the bias term, θ_0. I.e. the θ vector is indexed from 0 to n (holding $n+1$ values, θ_0 through θ_n), and this sum explicitly skips θ_0, by running from 1 to n, skipping 0.

Gradient Descent:

Just like with linear regression, we will want to separately update θ_0 and the rest of the parameters because we do not want to regularize θ_0.

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right] \quad j \in \{1, 2 \ldots n\}$$

}

This is identical to the gradient descent function presented for linear regression.

~~Emitted Code Feature Vector~~

Regularized Logistic Regression: Video: