Week 3

**Classification and Representation**

**Classification:**

* Don’t use linear regression for classification problems

**Hypothesis Representation:**

* Sigmoid/Logistic Function
  + hθ(x) = g(θTx)
  + g(z) = 1/(1+e–z)
  + hθ(x) = 1/(1+e–(theta^T)x)
* hθ(x) = estimated probability that y = 1 on input x
* hθ(x) = p(y=1 l x ; 0) -> probability that y = 1, given x, parameterized by θ
* p(y=0 l x ; θ) + p(y= l x ; θ) = 1

**Decision Boundary**

* predict y = 1 if hθ(x) >= 0.5 🡪 θTx >= 0
* predict y = 0 if hθ(x) < 0.5 🡪 θTx < 0

**Cost Function**

* Cost(hθ(x), y) =
  + – log(hθ(x)) if y = 1
  + – log(1 – hθ(x)) if y = 0
* Cost = 0 if y = 1 and hθ(x) = 1
* As hθ(x) 🡪 0, Cost 🡪 infinity
* Captures intuition that if hθ(x) = 0 (predict P(y = 1 l x ; θ), but y = 1, we’ll penalize learning algorithm by a very large cost

**Logistic Regression Cost Function**

* Cost(hθ(x), y) = – ylog(hθ(x)) – (1 – y)log(1 – hθ(x))
  + More compact way of writing cost function when y = 1 and y = 0
* Gradient Descent
  + J(θ) = –(1/m) [ summation i = 1 to m ] (y(i)log(hθ(x(i))) + (1 – y(i))log(1 – hθ(x(i)))
  + Want minθ J(θ)
    - Algorithm looks identical linear regression for repeat
    - Repeat θ**j :=** θ**j –** α(summation i = 1 to m)( hθ(xi) – yi )xji (simultaneously update (θj for j = 0, … n)
    - thing changed is hθ(x), which is now (1/(1+e-thetaTx) instead of θTx

**Advanced Optimization**

* Optimization algorithms:
  + Gradient descent
  + Conjugate descent
  + BFGS
  + L-BFGS
* Advantages:
  + No need to manually pick α
  + Often faster than gradient descent
* Disadvantages:
  + More complex

**Multiclass Classification: One-vs-all**

* Email foldering/tagging: Work, Friends, Family, Hobbies …
* Medical diagrams: Not ill, Cold, Flu
* Weather: Sunny, Cloudy, Rain, Snow
* Train a logistic regression classifier hθ(i)(x) for each class i to predict the probability that y = i

**The Problem of Overfitting**

* Underfit and high bias
* Just right
* Overfit and high variance
* Overfitting:
  + If we have too many features, the learned hypothesis may fit the training set very well, but fail to generalize to new examples.
* Addressing overfitting
  + Reduce number of features
    - Manually select which features t keep
    - Model selection algorithm
* Regularization
  + Keep all the features, but reduce magnitude/values of parameters θj
  + Works well when we have a lot of features, each of which contributes a bit to predicting y.

**Cost Function**

* Regularization:
  + Small values for parameters
    - Simpler hypothesis
    - Less prone to overfitting
* ***J(***θ*) = (1/2m){[(summation i = 1 to m)([h*θ(x) – y)]2 + λ(summation j = 1 to n)(θ2j)}
  + λ is regularization parameter
  + if λ is too large, then algorithm underfits

**Regularized Linear Regression**