**Binary Classification**

|  |
| --- |
| **Input and output** |
| **Input:**  M example, n+1 features, x0(i) are 1’s  X = m\*n matrix  Y = m\*1 matrix, valued {0, 1}  **output**  Predict y, where y∈{0,1} (binary classification problem)  0 = Negative class  1 = positive class |
| **Logistic Regression (Classification Algorithm)** |
| **0 <= hθ(x) <= 1**  Logistic Regression Formula:  Sigmoid function:  Combined:  **Probability**  h(x) give the probability that y = 1 (0.7 = 70% to be 1)  **Probability Notation**  **hθ(x) = P(y = 1 | x; θ)** #probability that y = 1, given x, is parameterized by θ  **P(y = 1 | x; θ) + P(y = 0 | x; θ) = 1** |
| **Decision Boundary** |
| If **hθ(x) ≥ 0.5:**  y = 1  if **hθ(x) < 0.5:**  y = 0  if **z ≥ 0** or  **≥ 0**:  g(z) ≥ 0.5  y = 1  if **z < 0 or < 0** :  g(z) < 0.5  y = 0  **Decision Boundary -** The line that separates the area where y = 0 and where y = 1    **In this case,**  The decision boundary is a vertical line, where everything to the left denote y = 1  Note:  z could be any function or even a circle |
| **Convex vs Non-Convex** |
| |  |  | | --- | --- | | Non-convex function | Convex function | | >1 local optimum |  | |
| **Cost Function** |
| **Simplified Cost Function**       |  |  | | --- | --- | |  |  |   **Note:**  If h(x) == y:  Cost(h(x), y) = 0  If h(x) == 0 and y == 1:  Cost(h(x), y) = error 🡪 inf  If h(x) == 1 and y = 0:  Cost(h(x), y) = error 🡪 inf  Penalize learning algo. With a large number if the prediction is very off. |
| **Classification Gradient Descent** |
| **Update Parameters** |
| **Advanced Optimization** |
| function [jVal, gradient] = costFunction(theta)  jVal =  gradient =  end |
| options = optimset('GradObj', 'on', 'MaxIter', 100); %gradient descent provided ‘on’, max iteration = 100  initialTheta = zeros(2,1);  [optTheta, functionVal, exitFlag] = fminunc(@costFunction, initialTheta, options);  **Note:**  % optTheta returns the optimum parameter value  % FunctionVal = slope of tangent line at minimum of J(θ)  % exitFlag shows whether the learning algorithm has converged  % fminuc function only work when size(θ, 1) ≥ 2 |

**Multiclass Classification**

|  |
| --- |
| **Idea** |
| **Predictions**    **Pick Prediction**  **For each training case, the training case (i) that predict the highest value, is the class where the data lay.**    Each classifier h(x)(i) for each class predict the probability that y = i |

Problem of overfitting

|  |
| --- |
| **Explanation** |
| |  |  |  | | --- | --- | --- | | Underfitting: “high bias” too little variable over training set | Good example | Overfitting: too much variable over training set.  Unlikely to generalize well for new example | |  |  |  |   **Ways to resolve overfitting**   1. **Reduce #features**   Select which features to keep  Model selection algorithm   1. **Regularization**   Keep all features, reduce magnitude of θ  Small values for parameter θ, gives smoother, generalized, and simpler function  Each of which contributes to a bit to predicting y |

|  |
| --- |
| **Regularization – Linear Regression** |
|  |
| **Modify Cost Function - example**    *In this example, in order to minimize the cost, θ3, 4 need to be small to do so*  **Modified Cost Function formula**    *θ0 are not penalized*  λ control a constant that would keep the fitting of parameters well (the square difference) and keep the parameters small (the lambda term)  If λ is too large, it will results in underfitting |
| **Regularization – Gradient descent & Normal Equation** |
| **Modified Gradient descent with respect to modified cost function**    Or  Since  must not <= 0, therefore  must be < 1, makes theta j a bit smaller  **Normal Equation**  where L is a (n+1) \* (n+1) matrix |

|  |
| --- |
| **Regularization – Logistic regression** |
| Regularized Cost Function |
| Where theta is squared  **Gradient descent** |