**Week 3**

* **Classification and Representation**
  + **Usually, applying linear regression to classification problem is not a good idea, especially when there are extreme values which would distort the model.**

y = 0 or 1 but can be >1 or <0 in a linear regression model.

* + Chart, line chart

    Description automatically generatedA picture containing diagram

    Description automatically generated**Hypothesis representation: sigmoid function/logistic function (binary, y = 1 or 0)**

**\*Transform from domain (-∞, ∞) to (0, 1) to signify the probability of y = 1.**

* + **Decision boundary:**

**A property of hypothesis about parameters (θ), not about the dataset.**

The function can be very complex in order to form a desirable shape to fit the dataset.

* Logistic Regression Model:
  + Cost function:

Non-convex: local minimum

Convex: one local minimum (global minimum)

A picture containing text, clock, watch, gauge

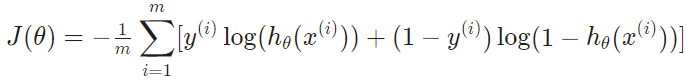
Description automatically generatedIf we use the cost function for linear regression, it will result in non-convex funct ion:

Text, letter

Description automatically generatedTherefore, we must transfer the cost function into a convex function to better conduct gradient descent:

\*Statistically maximum likelihood

**Entire cost function:**



**A picture containing text

Description automatically generatedVectorized implementation:**

**Text, letter

Description automatically generatedGradient descent:**

* + Optimization algorithm:
    - Gradient descent
    - Conjugate gradient
    - BFGS
    - L-BFGS

Advantages of other algorithm:

* + - No need to manually pick α
    - Often faster than gradient descent

Disadvantage:

* + - More complex
  + Implementation function:

function [jVal, gradient] = costFunction(theta)

jVal = [...code to compute J(theta)...];

gradient = [...code to compute derivative of J(theta)...];

end

options = optimset('GradObj', 'on', 'MaxIter', 100);

initialTheta = zeros(2,1);

[optTheta, functionVal, exitFlag] = fminunc(@costFunction, initialTheta, options);

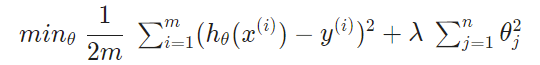
* Multiclass classification:
  + Text, letter

    Description automatically generatedOne-vs-all classification strategy:

Number of hypothesis = number of classes

* **Solving the problem of overfitting (regularization):**
  + **Overfitting**: high variance, too many features. Likely to fail to predict new examples.
  + Underfitting: high bias, too few features.
  + Addressing overfitting:
    1. Reduce number of features:
       - Manually select which features to keep
       - Model selection algorithm
    2. Regularization:
       - Keep all the features, but reduce magnitude/values of parameters .
       - Works well when we have a lot of features and each of them contributes a bit to predicting y.
  + Regularization:

Penalize some parameters in the cost function model.

* + - Benefits:
      * Simpler hypothesis (can be seen as reducing penalized parameters)
      * Smoothers fits
      * Less prone to overfitting.
  + **Regularized Cost function:**

\*we do not penalize

* + - λ: regularization parameter

Lambda determines how much the costs of our theta parameter are inflated in the cost function. If lambda is too large, it will result in underfitting. If lambda is too small, the regularization will not have much effect.

* + **Regularized linear regression:**
    - **Text, letter

      Description automatically generatedGradient descent:**
    - **Diagram

      Description automatically generatedNormal equation:**

Non-invertibility: if m < n, then is non-invertible. However, adding the term λ⋅L makes the whole matrix invertible.

* + Text

    Description automatically generatedRegularized Logistic Regression:

Cost function关于θJ的导数和linear regression相同，可证明，已证明。关键在于：

A picture containing diagram

Description automatically generated