Machine Learning Continued

by Jake Sauter

Logistic Regression

- Logistic Regression is one of the most popular and widely used classification algorithms
- It can be applied in situations in which the desired prediction is a discrete class (spam / not spam ; malignant / benign)
- It is called logistic regression as it makes use of the **logistic function** also known as the **sigmoid function**

Logistic Regression

- Logistic Regression must be used in these situations where we would like discrete class output, as if we used the linear regression prediction model our class bounds can be exceeded
- The hypothesis function $h_\theta(x)$, where θ is the vector of parameters fit to the model of the training data is usually (in linear regression) described as

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

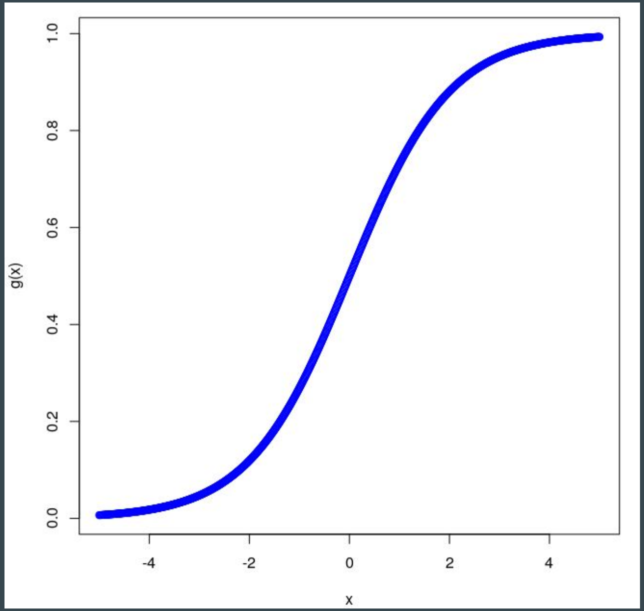
Logistic Regression

- A slight difference is implemented in logistic regression, letting the hypothesis function be

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

where ***g(x)*** is the sigmoid function defined as

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



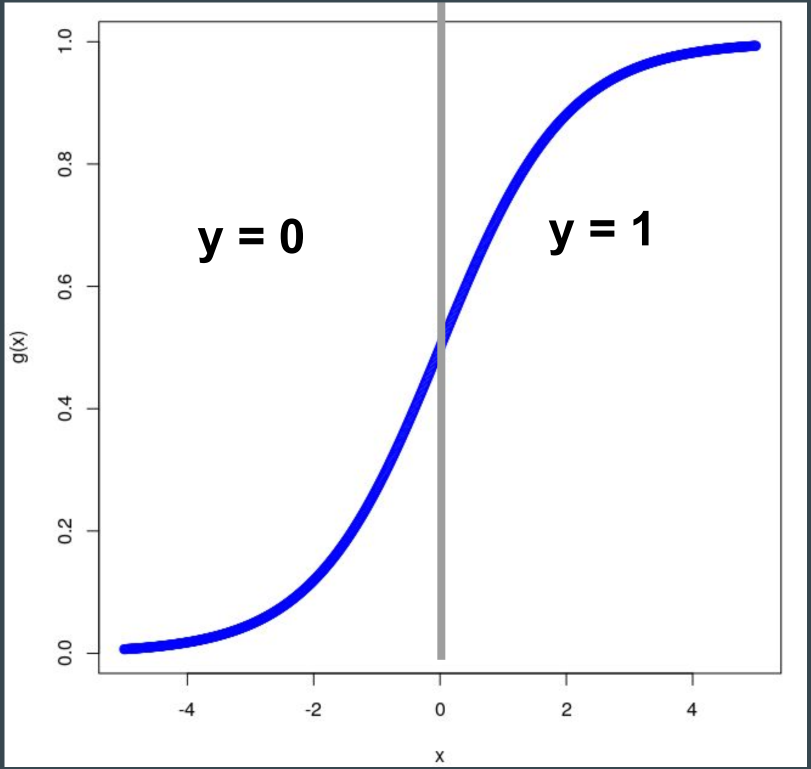
Logistic Regression

- The interpretation of the output of $h_\theta(x)$ is the probability of the sample with feature vector x belonging to the positive class $(y = 1)$

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

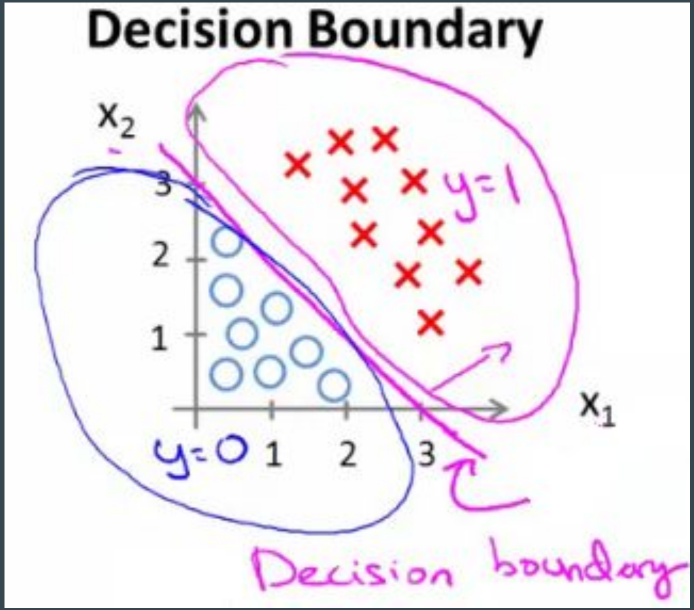
- The **decision boundary** of the hypothesis can be intuitively derived
  - We need to decide when we will predict $y = 1$ and $y = 0$, an intuitive decision could be assigning y to 1 if $h_\theta(x) > 0.5$ and y to 0 if $h_\theta(x) \leq 0.5$
  - Since $g(x) = 0.5$ at $x = 0$, these conditions can be written as $y = 1$ when $\Theta^T(x) \geq 0$ and $y = 0$ when $\Theta^T(x) \leq 0$

Logistic Regression Visualized



Logistic Regression: Linear Decision Boundaries

- Simple implementations of logistic regression implement a **linear decision boundary**, which in terms of our parameter vector and hypothesis simply means that the parameter coefficients for each given feature are linear

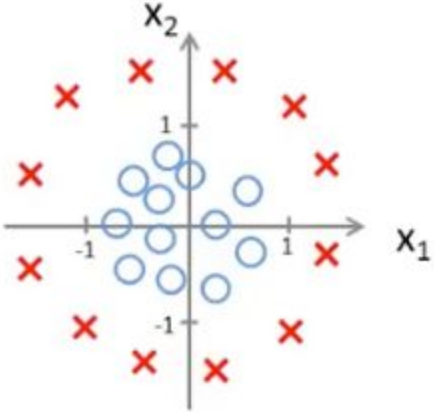


[2]

Logistic Regression: Non-Linear Decision Boundary

- Since in logistic regression we do not limit the form of this feature vector, we may also introduce non-linear terms to form a **non-linear decision boundary**

Non-linear decision boundaries



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

[2]

Logistic Regression : Fitting Parameters

- So far logistic regression has seemed to be intuitive, though we have not seen how to actually calculate the parameter vector $\boldsymbol{\theta}$ to make the decisions that we have prompted
- Briefly, a **cost function** is a function that provided a **training set** for fitting model parameters, will provide us with how well the model's predictions match the true class of each training example
- We must begin with the **cost function** of linear regression
  - In simple linear regression, we defined the cost function to be

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Logistic Regression: Cost Function

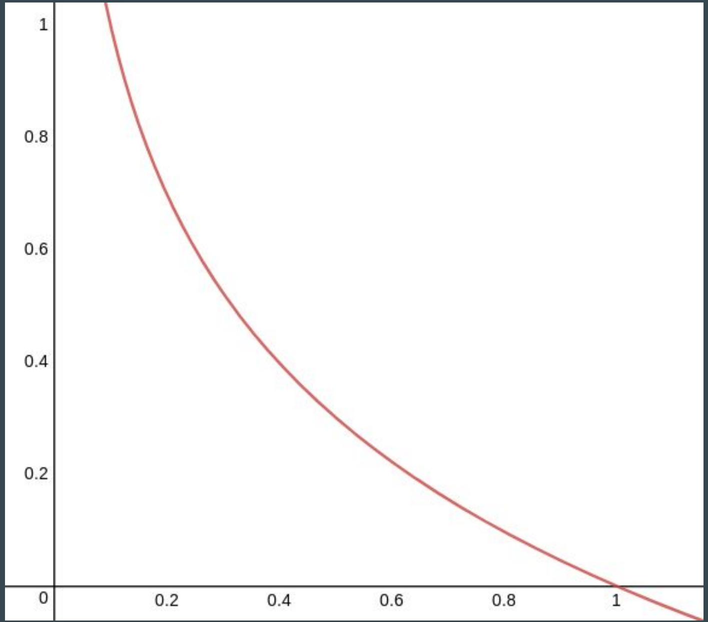
- Though this is not ideal for us as this is a non-convex cost function with respect to our new hypothesis function, meaning that **gradient descent** (our method of finding parameters θ) will not work
- To adapt this cost function into a convex function for logistic regression, we can begin by splitting the cost function into two cases

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)); y = 1 \\ -log(1 - h_\theta(x)); y = 0 \end{cases}$$

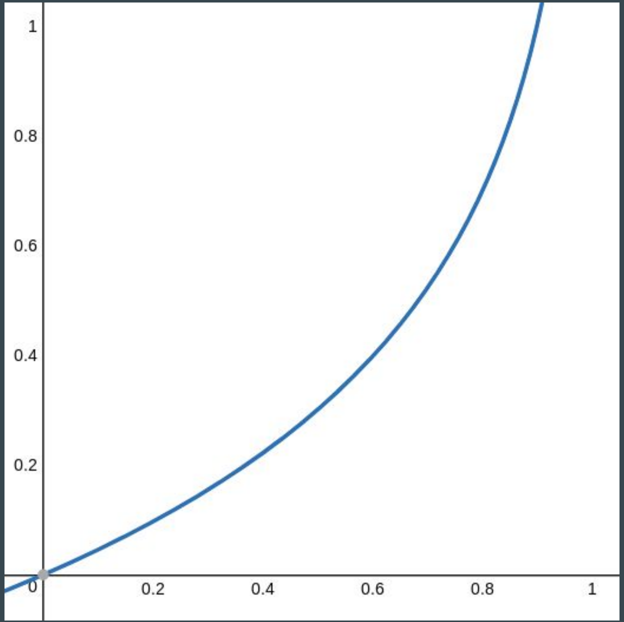
Logistic Regression

- Graphically, these two functions look like

If y=1



If y=0



Logistic Regression: Cost Function

- These two functions can be written in a single form

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

  as ***y*** strictly takes on the values 0 and 1 in the two possible cases of class membership

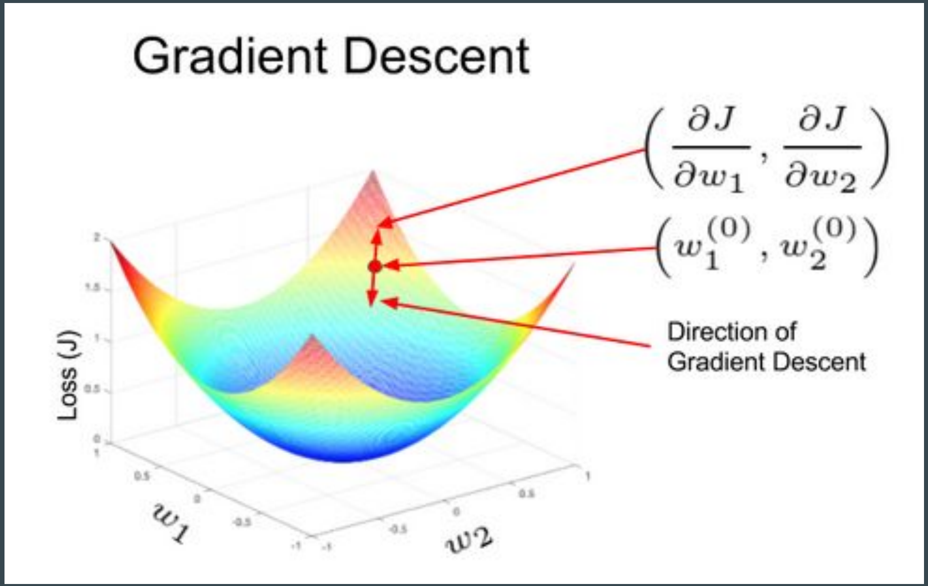
- This cost function can be derived from Maximum Likelihood Estimation, though also intuitively has many desirable properties for a cost function

Logistic Regression: Cost Function

- Properties of the cost function:
  - Convex for the logistic hypothesis function
  - Tends towards infinity as predicted class gets closer to incorrect class
  - Is 0 when predicted class is the correct class

Logistic Regression: Finding *Θ*

- How can we use the cost function that we have obtained to generate the optimal parameters for prediction?
  - Gradient Descent
- **Gradient Descent** is a repetitive process that can be used to find the minimum value of a function, and and which point in the native space this minimum occurs



[1]

Logistic Regression: Gradient Descent

- We can apply gradient descent to find the parameter vector $\boldsymbol{\theta}$ in the following way

$$
\begin{aligned}
&\text{Repeat}\{ \\
&\qquad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\
&\}
\end{aligned}
$$

where α is the **learning rate**, which is simply how far in the gradient direction we will step at each iteration. If α is too large, we may overstep the minimum and if α is too small, convergence may take a very long time

Gradient Descent

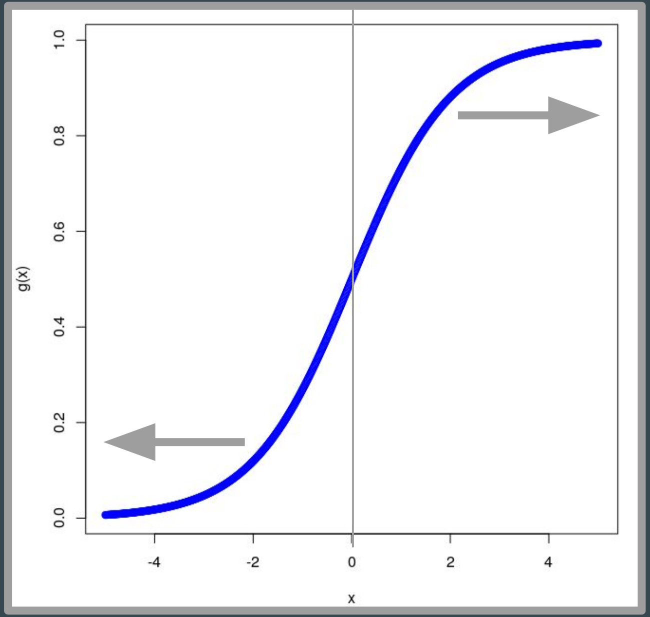
- One may note that this update rule is identical to that of linear regression, with the only difference being the hypothesis function
- Another important note is that advanced optimization techniques exist that can automatically select the learning rate such as
  - Conjugate Gradient
  - BFGS
  - L-BFGS
- One final note is that logistic regression can be used to predict multiclass problems by developing a model for each class and using them as discriminant functions

Support Vector Machines

- Support Vector Machines (or SVMs) can sometimes provide a cleaner and more powerful way of learning complex non-linear functions compared to logistic regression and neural networks
- In reviewing the SVM, we can actually see it as a modification of logistic regression, in which we can estimate the cost function in a simpler way and obtain a larger classification margin

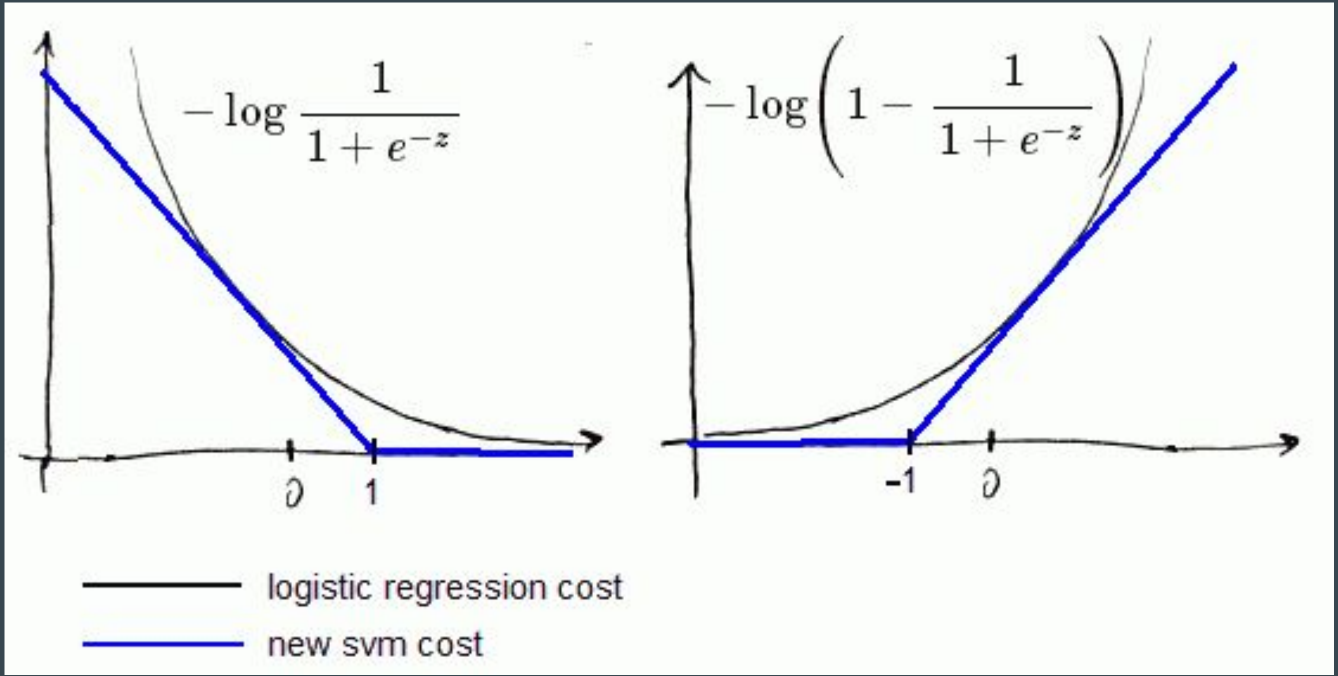
Support Vector Machines

- If we view logistic regression in the light of constructing a large margin classifier, if y=1, we want $h_\theta \simeq 0$, which implies that $\theta^T x >> 0$



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

Support Vector Machine: Cost Function Estimation



[2]

Support Vector Machine: Cost Functions

- These cost functions add a margin to classification, as the cost is not 0 just when the example is classified correctly, but when it is classified correctly by a large margin (a margin of 1)
- This characteristic is what makes SVMs a **large margin classifier**

Aside: Regularization

- **Regularization** refers to the process of scaling features to avoid overfitting, which can be facilitated by adding a regularization term to the cost function
- This will provide an incentive for lower feature scalings as the magnitude of features will directly influence the cost function
- Often the added regularization term is of the form $\lambda \sum_{i=1}^{m} \theta_j^2$ where λ is the regularization parameter

SVM: Cost function

Logistic Regression cost function:

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

$$\Downarrow$$

$$J(\theta) = \frac{1}{m}[\sum_{i=1}^{m} y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)})\mathrm{cost}_0(\theta^T x^{(i)})] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

SVM Cost Function

- Usual modifications to this cost function include
    - Reparameterizing the regularization coefficient from A + λ B to CA + B
    - dropping all **1/m** terms for simplification

$$J(\theta) = C[\sum_{i=1}^{m} y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)})\mathrm{cost}_0(\theta^T x^{(i)})] + \frac{1}{2}\sum_{j=1}^{n} \theta_j^2$$

Support Vector Machines: Output Interpretation

- SVMs do not output probabilities, but output predictions

$$h_\theta(x) = \begin{cases} 1 \text{ if } \theta^T x \geq 0 \\ 0 \ \ \text{otherwise} \end{cases}$$

Support Vector Machine: Mathematical Intuition

- So how does this optimization objective lead to our classifier having a large margin?
- First we must modify the cost function

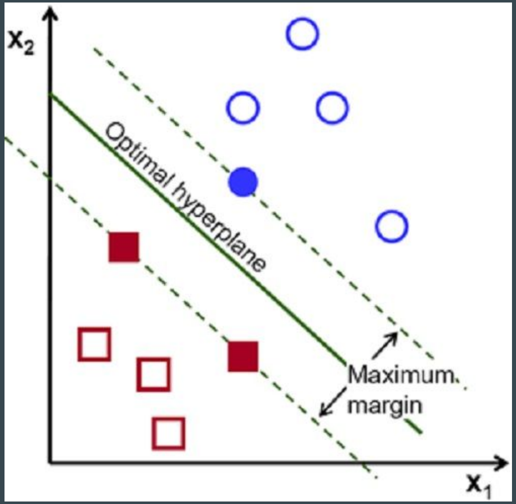
$$J(\theta) = \frac{1}{m}[\sum_{i=1}^{m} y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)})\text{cost}_0(\theta^T x^{(i)})] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

$$\Downarrow$$

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}\theta_j^2 \quad s.t. \quad \begin{array}{l} \theta^T x^{(i)} \geq 1 \text{ if } y^{(i)} = 1 \\ \theta^T x^{(i)} \leq -1 \text{ if } y^{(i)} = 0 \end{array}$$

Support Vector Machine: Mathematical Intuition

- Now to obtain some geometric intuition of how this optimization objective translates to the standard image in explanation of SVMs, we will perform a simple conversion



[2]

$$J(\theta) = \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 = \frac{1}{2}(\theta_1^2 + \theta_2^2 + \cdots \theta_n^2) = \frac{1}{2}(\sqrt{\theta_1^2 + \theta_2^2 + \cdots \theta_n^2} = \frac{1}{2}||\theta||^2$$

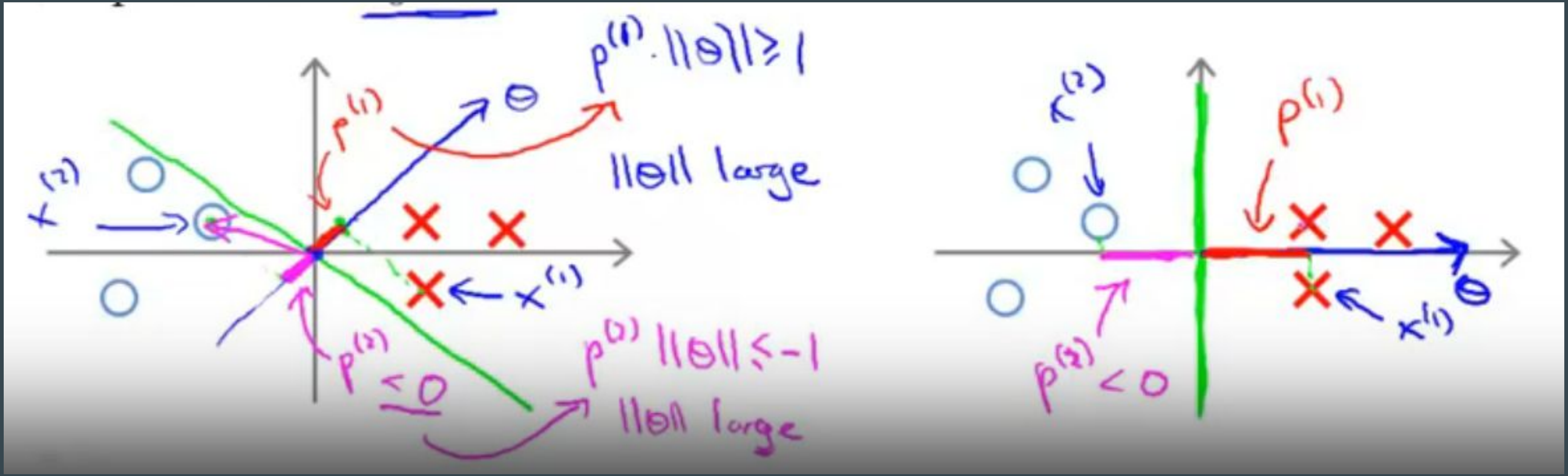
Support Vector Machine: Mathematical Intuition

- We will also make a modification to the restraint conditions that must be met, noting that $\theta^T x^{(i)} = p^{(i)}||\theta||$, we now have

$$J(\theta) = \frac{1}{2}||\theta||^2 \quad \text{s.t.} \quad \begin{array}{l} p^{(i)}||\theta|| \geq 1 \text{ if } y^{(i)} = 1 \\ p^{(i)}||\theta|| \leq -1 \text{ if } y^{(i)} = 0 \end{array}$$

- With this alternate view of the cost function, we can see that the optimization objection is really to minimize the squared norm of the parameter vector ***Θ*, which** can only be done by maximizing the projection of each training example onto ***Θ***

Support Vector Machine: Mathematical Intuition



[2]

Briefly: Random Forests

- **Random forests** have been developed to combat many common issues in decision trees, primarily overfitting
    - They reduce overfitting without substantially increasing error due to bias (which is usually introduced when limiting depth to prevent overfitting)
- To produce a more robust classifier, random forests build many standard decision trees on random subsets of the data, and only use randomly selected features of each of these subsets
    - For final classification of new samples, each tree is allotted a vote on the true class, and the predicted class of the model is the class with the largest overall sum of votes

Applications

- All of the discussed supervised classifiers can be easily implemented in R, with the following results for the first 5 principle components of SAM selected differentially expressed genes with a median fdr of .05

```
           knn_output
test_labels ALL AML
        ALL  19   1
        AML   2  12
```

```
           test_labels
tree_output ALL AML
        ALL  19   2
        AML   1  12
```

```
             test_labels
forest_output ALL AML
          ALL  20   2
          AML   0  12
```

```
           svm_output
test_labels ALL AML
        ALL  19   1
        AML   1  13
```

```
          test_labels
log_output ALL AML
       ALL  19   1
       AML   1  13
```

References

[1] Sharma, Aditya. “Understanding Activation Functions in Deep Learning.” Learn OpenCV, Learnopencv, 30 Oct. 2017, www.learnopencv.com/understanding-activation-functions-in-deep-learning/.

[2] “Machine Learning.” Coursera, Standford University, 2018, www.coursera.org/learn/machine-learning.

[3] Raval, Siraj. “Random Forests - The Math of Intelligence (Week 6).” YouTube, YouTube, 26 July 2017, www.youtube.com/watch?v=QHOazyP-YlM.

Question: Gradient Descent

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