Logistic Regression

Swakkhar Shatabda

CSE 489: Machine Learning, Summer 2017
Department of Computer Science and Engineering
United International University

Classification

1. In classification problems, we are given data as X and labels as y.
2. Here, we are upto learn a model where, y will be predicted as a function of X.
3. In classification, the label y is categorical or discrete in value.
4. For example, suppose you are given many features of a fish, like length, weight, eggs, months and you have to predict whether it is legal to be caught or not. This problem can be formulated as a classification problem.

Data

Here is how data looks like in a supervised setting:

	features				label
	x_1	x_2	x_3	x_4	y
instance no	length	weight	has eggs	month	legal?
1	10	250	1	12	No
2	20	1250	0	1	Yes
3	15	750	1	2	No
⋮	⋮	⋮	⋮	⋮	⋮
m	17	550	0	3	Yes

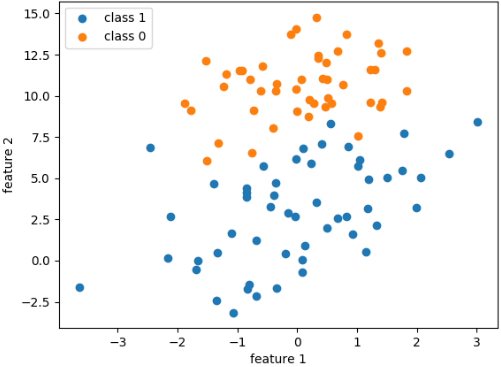
Experiments

We will first try to predict the class of the dataset based on two features, x_1 and x_2.

```
-0.017612,14.053064,0
-1.395634,4.662541,1
-0.752157,6.53862,0
-1.322371,7.152853,0
0.423363,11.054677,0
0.406704,7.067335,1
0.667394,12.741452,0
-2.46015,6.866805,1
0.569411,9.548755,0
-0.026632,10.427743,0
0.850433,6.920334,1
1.347183,13.1755,0
1.176813,3.16702,1
-1.781871,9.097953,0
-0.566606,5.749003,1
0.931635,1.589505,1
-0.024205,6.151823,1
-0.036453,2.690988,1
-0.196949,0.444165,1
1.014459,5.754399,1
1.985298,3.230619,1
-1.693453,-0.55754,1
```

Experiments

We will first try to predict the class of the dataset based on two features, x_1 and x_2.



Logistic Regression

At first, we are going to try a linear classifier called logistic regression. We can apply logistic regression when the data is linearly separable.

- The relationship will be predicted as:

$$y = w_0 + w_1 x_1$$

- This is again an equation of a straight line
- We need the best line that separates blue from the orange
- learn $w_0, w_1, \cdots$
- Can we use gradient descent here? A little trick required!

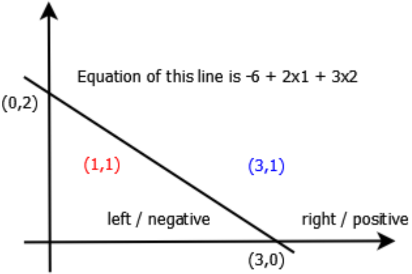
Gradient Descent for Logistic Regression

- The cost function / loss function of gradient descent

$$e = \frac{1}{2}\sum_{i=1}^{m}(\hat{y}(i) - y(i))^2$$

- This time too predicted label $\hat{y}$ is a function of $\vec{x}$ and w
- The labels are discrete, for this binary classification two labels 0 (no or negative) and 1 (yes or positive)
- Now, we try to define $\hat{y}$ with help of the weights or coefficients of the line.

Linear Classification



- This linear classifier divides instances based on the local wrt the line, on the right positive, negative on the left
- Any point on the line satisfies the equation. Any point on the right (3,1) yields positive result and any point on the left (1,1) yields negative result.
- Based on this we can define a linear classifier

Linear Classification

This following function will help us in making decision:

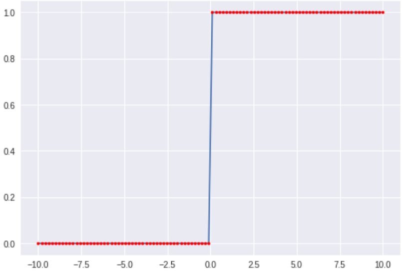
$$f(\vec{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

LinearClassifier

if $f(\vec{x}) > 0$ or $w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0$
    return 1
else return 0

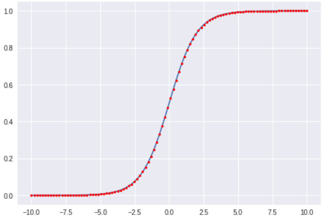
This simple classifier just checks whether a point is on the left or right.

A step function!



Alas! This is not a continuous function and thus not differentiable. We can't calculate gradients! We need to find an alternate!

A sigmoid function!



$$\sigma(\vec{x}) = \frac{1}{1+exp(-\vec{x})}$$

Good things about sigmoid!

1. Its continuous and differentiable.
2. $\sigma'(\vec{x}) = \sigma(\vec{x})(1 - \sigma(\vec{x}))$

Lets go back to the loss function now.

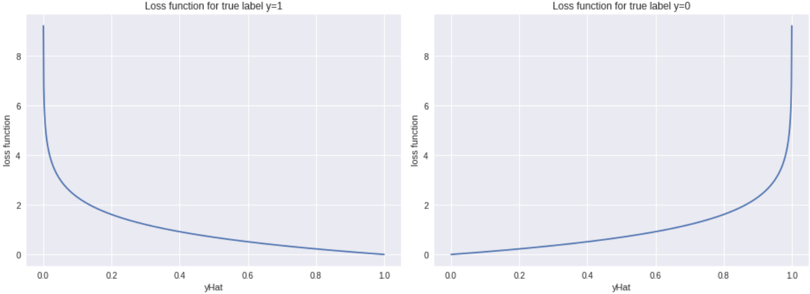
A new loss function - Cross-entropy

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.

$$e = \sum_{i=1}^{m} (-y(i) log(\hat{y}(i)) - (1 - y) log(1 - \hat{y}))$$

Cross Entropy Loss Function

How it works?



$$e = \sum_{i=1}^{m} (-y(i) log(\hat{y}(i)) - (1 - y) log(1 - \hat{y}))$$

Cross Entropy Loss Function

How to find the gradient? Lets try!

$$
\begin{aligned}
\frac{\delta e}{\delta w_0} &= \frac{\delta}{\delta w0} \sum_{i=1}^{m} (-y(i) log(\hat{y}(i)) - (1-y) log(1-\hat{y}(i))) \\
&= \sum_{i=1}^{m} (-y(i) \frac{1}{\hat{y}(i)} \hat{y}(i)(1-\hat{y}(i)).1 - (1-y) \frac{1}{(1-\hat{y}(i))} (-1)) \hat{y}(i)(1-\hat{y}(i)).1) \\
&= \sum_{i=1}^{m} (-y(i) + y(i)\hat{y}(i) + \hat{y}(i) - y(i)\hat{y}(i)).1 \\
&= \sum_{i=1}^{m} (\hat{y}(i) - y(i)).1
\end{aligned}
\tag{1}
$$

In a similar way,

$$\frac{\delta e}{\delta w_i} = \sum_{i=1}^{m} (\hat{y}(i) - y(i)).x_i \tag{2}$$

Now the same gradient descent will work!

Comments on Gradient Descent

1. Slow when the dataset is too large!
2. Rather learning the whole dataset, possible to learn in chunks!
3. What if we process only 1 single item at each iteration?
4. Lets have another look!

Gradient Descent Algorithm

$\textsc{GradientDescent}(X, y, alpha, maxIter)$

for $j = 1$ **to** m
    $x_0(j) = 1$
$w_0, w_1, \cdots, w_n$ initialized randomly
$iter = 0$
while $iter{+}{+} \leq maxIter$
    for $j = 0$ **to** n
        $slope_j = 0$
    for $i = 1$ **to** m
        $\hat{y} = w_0 + w_1 x_1(i) + w_2 x_2(i) + \cdots + w_n x_n(i)$
        $e = \hat{y} - y(i)$
        for $j = 0$ **to** n
            $slope_j = slope_j + e \times x_j(i)$
    for $j = 0$ **to** n
        $w_j = w_j - \alpha \times slope_j$
return $w_0, w_1, \cdots, w_n$

Lighter Gradient Descent Algorithm

LIGHTERGRADIENTDESCENT($X, y, alpha, maxIter$)

for $j = 1$ **to** m
    $x_0(j) = 1$
$w_0, w_1, \cdots, w_n$ initialized randomly
$iter = 0$
while $iter{+}{+} \leq maxIter$
    for $j = 0$ **to** n
        $slope_j = 0$
    $i = iter$
    $\hat{y} = w_0 + w_1 x_1(i) + w_2 x_2(i) + \cdots + w_n x_n(i)$
    $e = \hat{y} - y(i)$
    for $j = 0$ **to** n
        $slope_j = slope_j + e \times x_j(i)$
    for $j = 0$ **to** n
        $w_j = w_j - \alpha \times slope_j$
return $w_0, w_1, \cdots, w_n$

Observations

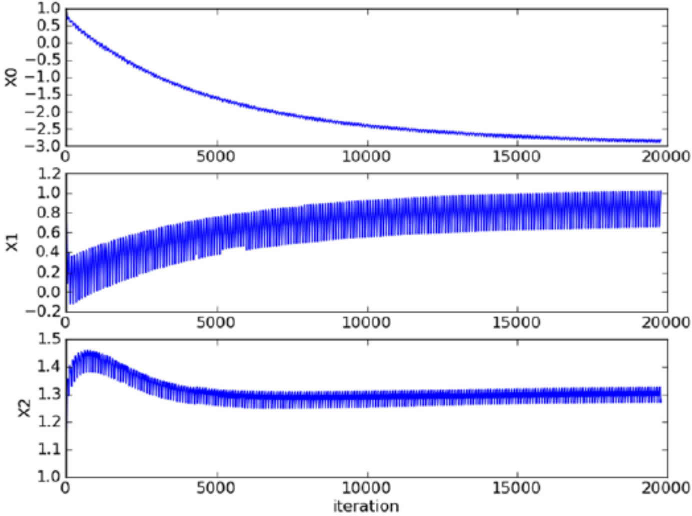


Figure 5.6 **Weights versus iteration number for one pass through the dataset, with this method. It takes a large number of cycles for the weights to reach a steady-state value, and there are still local fluctuations.**

Further Improvements

1. Decrease α
2. Shuffle dataset

Observations

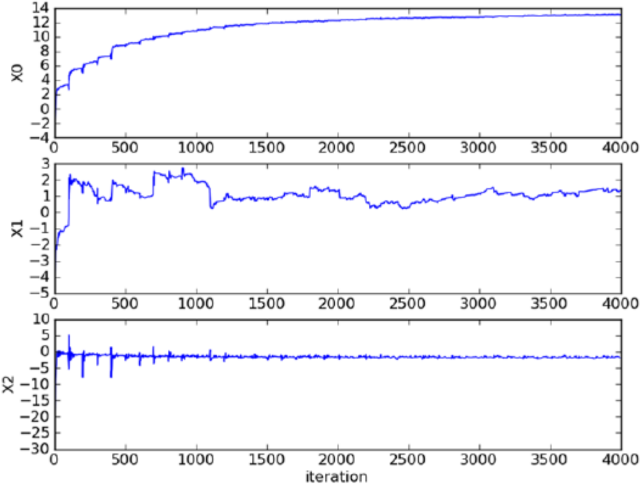


Figure 5.7 Coefficient convergence in `stocGradAscent1()` with random vector selection and decreasing alpha. This method is much faster to converge than using a fixed alpha.

Reading

Chapter 8, Machine Learning in Action
Chapter 18, Artificial Intelligence: A Modern Approach