3.24pt

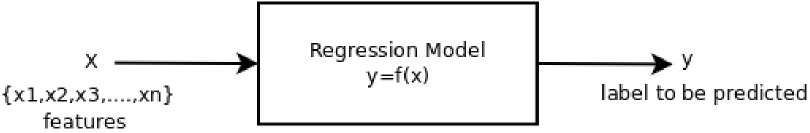
Linear Regression

Swakkhar Shatabda

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

Regression

1. In regression 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 regression, the label y is numeric and continuous in value.
4. For example, suppose you are given many features of a fish, like length, weight, eggs, months and you have to predict its price. This problem can be formulated as a regression 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	price
1	10	250	1	12	100
2	20	1250	0	1	500
3	15	750	1	2	1700
⋮	⋮	⋮	⋮	⋮	⋮
m	17	550	0	3	1000

Experiments

We will first try to predict the profit of a Food truck company earns from a city. Here the label to predict is the profit of the company (in \$10000s) in a month. That is our label, y. Now as feature first we will be considering only the population of the city (in 10000s), x_1.

ex0.csv

```
0.067732,3.176513
0.42781,3.816464
0.995731,4.550095
0.738336,4.256571
0.981083,4.560815
0.526171,3.929515
0.378887,3.52617
0.033859,3.156393
0.132791,3.110301
```

Linear Regression

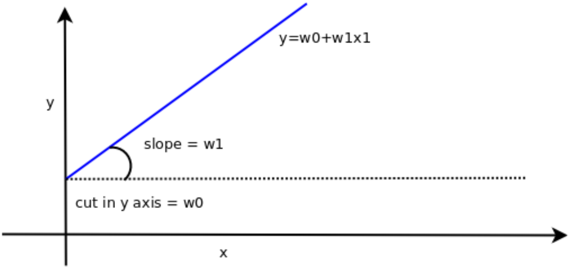
At first, we are going to try linear regression. Since y, profit depends only on one variable x_1, population, we call it **simple linear regression**.

1. The relationship will be predicted as:

$$y = w_0 + w_1 x_1$$

2. This is an equation of a straight line

Simple Linear Regression



- Here, w_0 is the value of y, when $x_1 = 0$, this could be either positive or negative
- Also note, w_1 is the rate of change of y, w.r.t. x_1
- We have to predict the relationship between x_1 and y, and we assume it to be linear.
- Here are problem is to estimate the correct values of w_0 and w_1

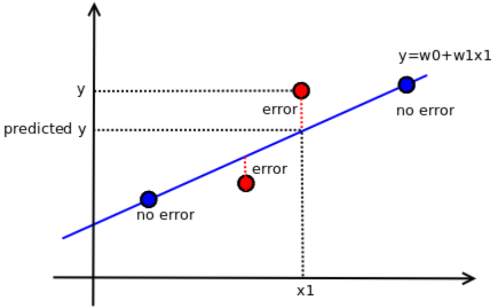
Error in Prediction

- We are going to formulate this estimation as an optimization problem, we will define an error and our goal will be to find such w_0, w_1 so that the error is minimized.
- The error function,

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

- Here, $\hat{y}(i)$ is the predicted label and $y(i)$ is the real label for a given instance or data i.
- We square it to negate the sign and put a half before for a mathematical convenience.

Explanation of Error



- Here the blue ones are correctly predicted by the line and thus have no error and the red ones are with errors.
- So error is the difference between real y and predicted $\hat{y} = w_0 + w_1 x_1$
- We can write,

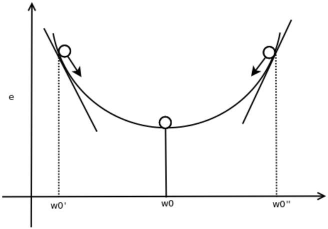
$$e = \frac{1}{2} \sum_{i=1}^{m} ((w_0 + w_1 x_1(i)) - y(i))^2$$

- Our task is to find w_0, w_1 so that the error e is minimized.

Finding w_0, w_1

- We call these co-efficients or weights.
- We are going to use gradient descent algorithm to find these values.
- The function is minimized at a point where the slope is zero.
- We randomly start from any value of w_0 or w_1 and eventually reach the minimum.
- Lets see how that is done!

Intution for Gradient Descent



- We could either start at w_0' or at w_0'' but we wish to reach w_0
- From w_0', we have to increase the value and move right and here at this point slope of the tangent is negative.
- From w_0'', we have to decrease the value and move left and here at this point slope of the tangent is positive.
- Its interesting to note that, more the distance from the point to the minimum is the value is slope is larger. We thus can change the weights proportionate to the slope.

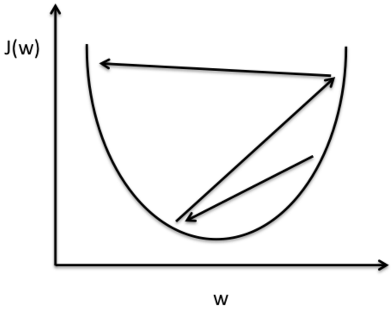
Intution for Gradient Descent

- However, a large value of slope might drastically change the value. We can minimize that effect by using a learning constant, α.
- Task of α is to control the changes of values of weights.
- The smaller the value of α is, slower the movement/change is. Again too high value will cause divergence.
- The increase or decrease in the values will be decided by the sign of the slope
- In general, we can apply the following in iterations:

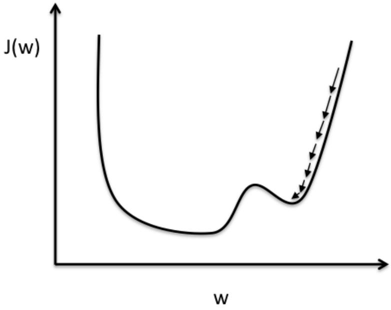
$$w_0(\text{new value}) = w_0(\text{old value}) - \alpha \frac{\delta e}{\delta w_0}$$

$$w_1(\text{new value}) = w_1(\text{old value}) - \alpha \frac{\delta e}{\delta w_1}$$

Learning Rate



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

Figure Soruce: `https://sebastianraschka.com/images/blog/2015/singlelayer_neural_networks_files/perceptron_learning_rate.png`

Finding Slopes

- To find slope we need to differentiate the following equation:

$$e = \frac{1}{2}\sum_{i=1}^{m}((w_0 + w_1 x_1(i)) - y(i))^2$$

- With respect to w_0,

$$\frac{\delta e}{\delta w_0} = \sum_{i=1}^{m}((w_0 + w_1 x_1(i)) - y(i)).1$$

- With respect to w_1,

$$\frac{\delta e}{\delta w_1} = \sum_{i=1}^{m}((w_0 + w_1 x_1(i)) - y(i)).x_1(i)$$

- Now, we will try to extend this for multi-variable linear regression:

$$\hat{y}(i) = w_0 + w_1 x_1(i) + w_2 x_2(i) + \cdots + w_n x_n(i)$$

- And, now we write a general equation for slope for a weight w_i:

$$\frac{\delta e}{\delta w_j} = \sum_{i=1}^{m}(\hat{y}(i) - y(i)).x_j(i)$$

- Note, each time the error in prediction is multiplied by 1 (for w_0) or multiplied by the feature x_j (for w_j)

Gradient Descent Algorithm

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 $w_0, w_1, \cdots, w_n$

Using linear algebra!

- Lets assume there are n features.
- Therefore, X becomes a matrix of $m \times n$
- And the array of labels, Y is a column vector $m \times 1$
- We add a single 1 in front all features of m instances and get a new matrix X_E with dimension, $m \times (n+1)$

$$\begin{vmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mn} \end{vmatrix}$$

Using linear algebra

- We assume the array of weights to be a column vector $W = [w_0 w_1 w_2 \cdots w_n]^T$ of dimension $(n+1) \times 1$
- Lets now multiply a row of X_E by W to get a single value which the prediction of y, $\hat{y}$ for that row.
- Thus if we multiple, the matrix X_E by W we get a column vector $m \times 1$, having all the estimations

$$\hat{Y} = X_E \times W$$

- Now, its very easy to find the error vector containing errors for all the predictions,

$$E = \hat{Y} - Y$$

- Note that each of the values of this vector E has to be multiplied by corresponding features and then summed.
- That is now easily achieved by the following multiplication:

$$S = X_E^T \times E$$

  here S is the array of the slopes.
- Finally, update weights:

$$W = W - \alpha \times S$$

Revised Gradient Descent

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

$X_E = [ones(m); X]$ \\ add 1s in front of all the rows
$W = [w_0, w_1, \cdots, w_n]$ initialized randomly
$iter = 0$
while $iter{+}{+} \leq maxIter$
    $\hat{Y} = X_E \times W$
    $E = \hat{Y} - Y$
    $S = X_E^T \times E$
    $W = W - \alpha \times S$
return $w_0, w_1, \cdots, w_n$

More Linear Algebra!

- We can do the whole thing without any loops!
- The error function is the trick!

$$
\begin{aligned}
e &= \sum_{i=1}^{m} (\hat{y}(i) - y(i))^2 \\
&= \sum_{i=1}^{m} (y(i) - \hat{y}(i))^2 \\
&= \sum_{i=1}^{m} (y(i) - x(i)W)^2 \\
&= (Y - XW)^T (Y - XW)
\end{aligned}
$$

- Now at minimum the differentiation of this error function must be 0.

$$
\begin{aligned}
\frac{\partial e}{\partial W} &= 0 \\
X^T (Y - XW) &= 0 \\
W &= (X^T X)^{-1} X^T Y
\end{aligned}
$$

- All weights to be found by a single line containing some matrix operations! (magic!)

Thats it!

Thank you