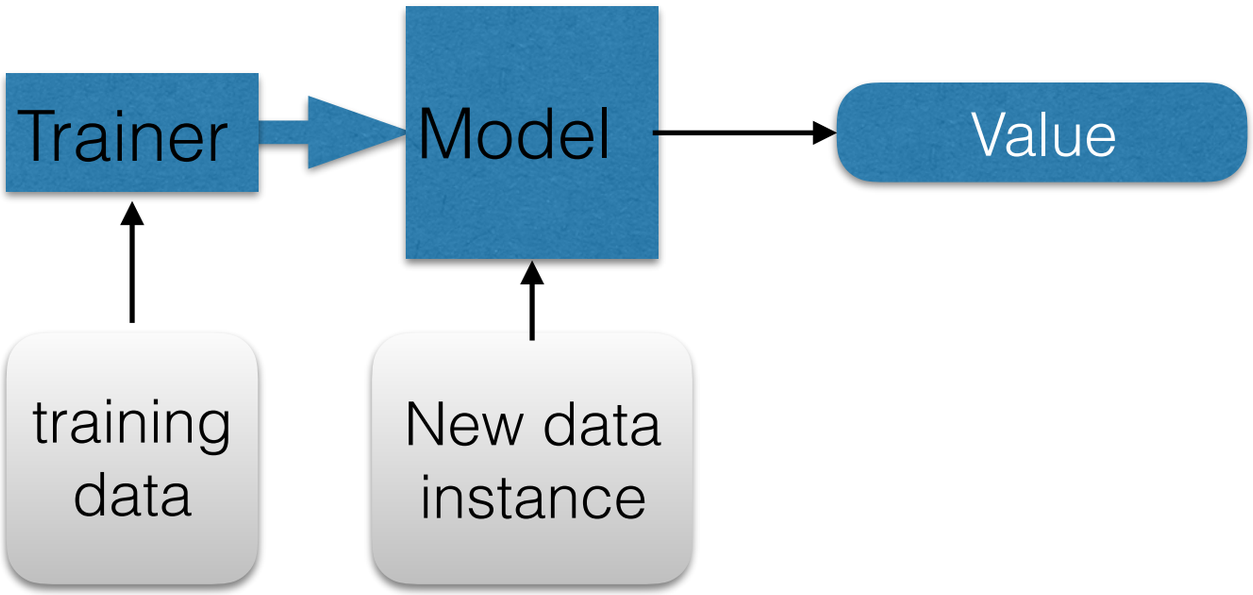
Linear Regression

Machine Learning

Objectives

- Students will be able to:
    - Describe how linear regression works.
    - Identify different cost functions
    - Know that for some cost functions we can solve for the minimum.
    - Know what the normal equation is.
    - Know that for some cost functions we need to search for the minimum.
    - Describe Gradient Descent
    - Describe Simulated Annealing
    - Discuss issues that can arise in a linear regression.
    - Discuss the performance of linear regression in terms of the number of features and the number of training instances.

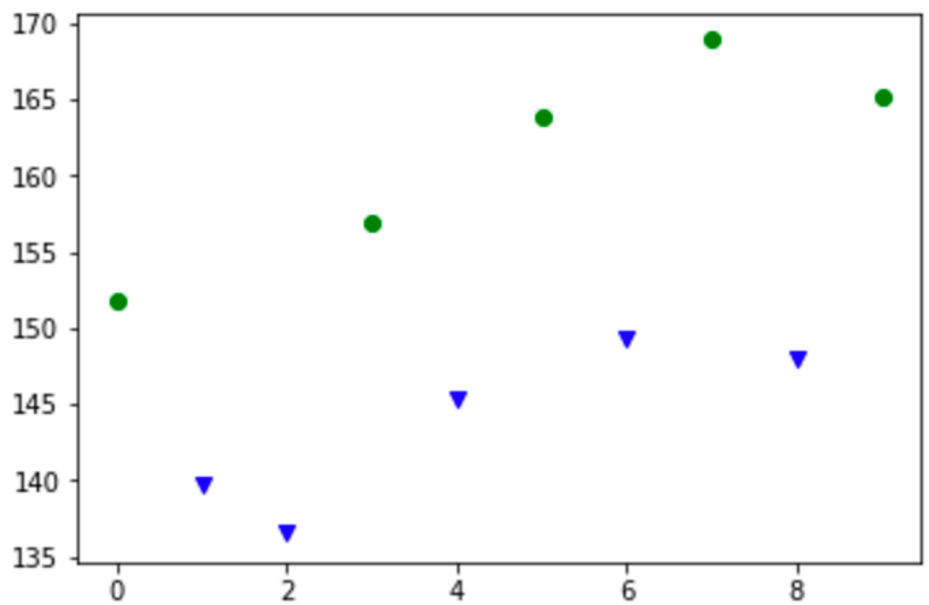
Linear Predictor



In linear regression, we will use data to a create a model that has a single numerical output.

Example

Suppose that we have the following data (from Howell) showing height vs weight.




```
(0, 151.765, 1)
(1, 139.7, 0)
(2, 136.525, 0)
(3, 156.845, 1)
(4, 145.415, 0)
(5, 163.83, 1)
(6, 149.225, 0)
(7, 168.91, 1)
(8, 147.955, 0)
(9, 165.1, 1)
```

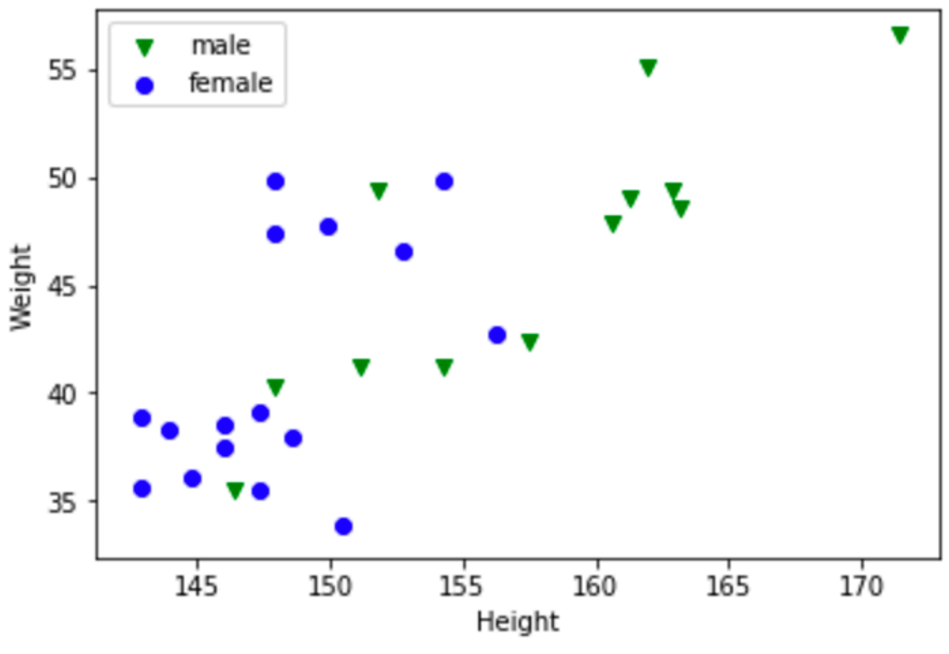
Example

We are going to create a model with a two parameters:

$$\theta_0 \quad \theta_1$$

To determine the output of the model we compute

$$weight = \theta_0 + \theta_1 \times height$$



The Linear Model

In a linear regression, we will sum the input values times a coefficient.

The model: $$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

- predicted value: $\hat{y}$
- bias: θ_0
- number of features: n
- feature values: $x_1, x_2, \ldots, x_n$
- model parameters: $\theta_0, \theta_1, \theta_2, \ldots, \theta_n$

The Linear Model (Vector Form)

In a linear regression, we will sum the input values times a coefficient.

The model: $$\hat{y} = h_\theta(\mathbf{x}) = \theta^T \mathbf{x}$$

- predicted value: $\hat{y}$

- parameter vector and transpose $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$ $\theta^T = [\theta_0, \theta_1, \theta_2, \ldots, \theta_n]$

feature vector (x0 is always set to 1) $\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

-

The Linear Model
Vector Form

When multiplying a matrix A times another matrix B, we compute the dot product of every row in A with every column in B.

Given: $\theta^T = [\theta_0, \theta_1, \theta_2, \ldots, \theta_n]$ $\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

The product is: $\theta^T \mathbf{x} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$

Evaluating the Model

There are a number of functions we can use to measure the performance, but there are a couple standard techniques.

1) For each data instance, compute a distance between the actual and the predicted values.

1. Absolute (L1 or Manhattan norm) $|h(\mathbf{x}^{(i)}) - y^{(i)}|$
2. Square (L2 or Euclidean norm) $(h(\mathbf{x}^{(i)}) - y^{(i)})^2$
3. Higher Powers (Lk norm) $(|h(\mathbf{x}^{(i)}) - y^{(i)}|)^k$

$\mathbf{x}^{(i)}$
input values for i'th data instance

$h(\mathbf{x}^{(i)})$
predicted value

$y^{(i)}$
actual value

2) Average over all instances

Norms: The Lk **norm** of a vector **v** is defined as follows. The higher the exponent, the more sensitive the norm is to outliers.

$$||\mathbf{v}||_k = \left(|v_1|^k + |v_2|^k + \ldots |v_n|^k \right)^{1/k}$$

Evaluating the Model
Mean Absolute Error (MAE)

We use the L1 norm and compute

$$MAE(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m} |\, h(\mathbf{x}^{(i)}) - y^{(i)} \,|$$

X is the training data and includes both $\mathbf{x}^{(i)}$ and $y^{(i)}$

h is the hypothesis function (model)

Evaluating the Model
Root Mean Square Error (RMSE)

We use the L2 norm and compute

$$RMSE(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}^{(i)}) - y^{(i)})^2}$$

X is the training data and includes both $\mathbf{x}^{(i)}$ and $y^{(i)}$

h is the hypothesis function (model)

Evaluating the Model
Mean Square Error (MSE)

We use the L2 norm and compute

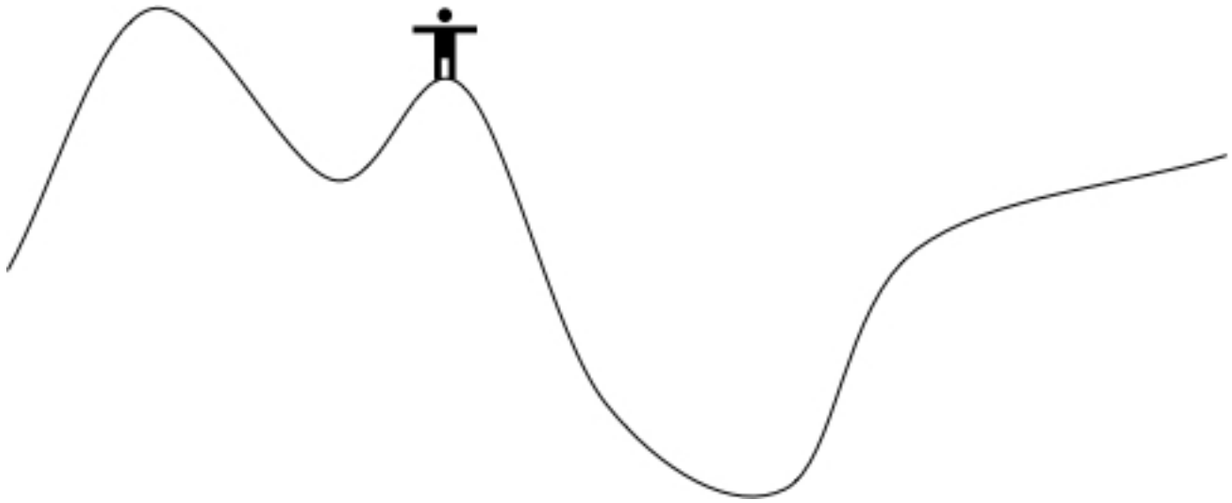
$$MSE(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}^{(i)}) - y^{(i)})^2$$

X is the training data and includes both $\mathbf{x}^{(i)}$ and $y^{(i)}$

h is the hypothesis function (model)

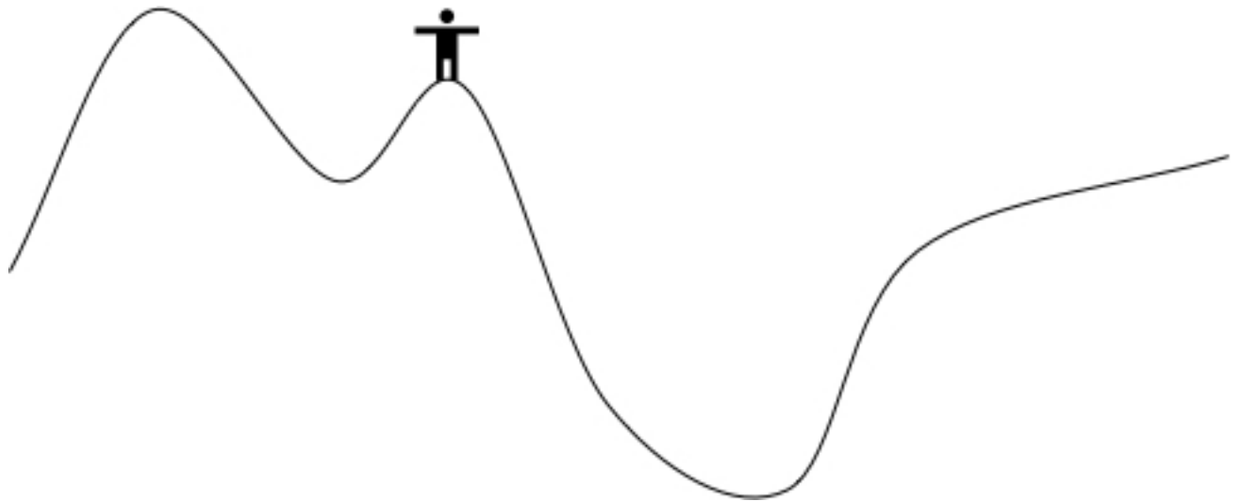
Minimizing a Function

Suppose that we want to find the minimum value of a function. What can you tell me about the shape of the curve at the minimum?



Minimizing a Function

At the minimum, the slope (derivative of the function) must be zero.
For a two (or higher) dimensional curve, the slope in all directions (gradient), must be zero.



Error-Revisit

For linear regression, using MSE results in an error function that is reasonably well behaved

- **MSE:** For linear regression, the error function is well behaved and if we compute the gradient we get linear equations that can be solved directly for the minimum. No search required.

- **RMSE**: The error function is well behaved, but the gradient does not result in linear equations. Use search.

- **MAE**: The error function is not well behaved. (Absolute value function does not have a derivative everywhere.) Use search.

Finding the gradient

To find the gradient, we need to compute a partial derivative for each of the regression parameters.

$$\frac{\partial}{\partial \theta_0}(\theta_0 + \theta_1 x^{(1)} - y^{(1)})^2 = 2(\theta_0 + \theta_1 x^{(1)} - y^{(1)})$$

$$\frac{\partial}{\partial \theta_1}(\theta_0 + \theta_1 x^{(1)} - y^{(1)})^2 = 2(\theta_0 + \theta_1 x^{(1)} - y^{(1)})(x^{(1)})$$

Notice that while the Error function is quadratic, the derivative is linear.

$$E = \frac{1}{3}\left\{(\theta_0 + \theta_1 x^{(1)} - y^{(1)})^2 + (\theta_0 + \theta_1 x^{(2)} - y^{(2)})^2 + (\theta_0 + \theta_1 x^{(3)} - y^{(3)})^2\right\}$$

Evaluating the Model

Lets see what happens when use MSE on a linear regression model that has 1 feature and 3 instances in the training data.

$$E = \frac{1}{3}(h(\mathbf{x}^{(1)}) - y^{(1)})^2 + (h(\mathbf{x}^{(2)}) - y^{(2)})^2 + (h(\mathbf{x}^{(3)}) - y^{(3)})^2)$$

$$E = \frac{1}{3}\left\{(\theta_0 + \theta_1 x^{(1)} - y^{(1)})^2 + (\theta_0 + \theta_1 x^{(2)} - y^{(2)})^2 + (\theta_0 + \theta_1 x^{(3)} - y^{(3)})^2\right\}$$

Our goal is to find the values for theta 0 and theta 1 that will minimize E.

Finding the gradient

Setting the derivative of E to zero gives two linear equations in two unknowns.

$$\frac{\partial E}{\partial \theta_0} = \frac{2}{3}\left\{(\theta_0 + \theta_1 x^{(1)} - y^{(1)}) + (\theta_0 + \theta_1 x^{(2)} - y^{(2)}) + (\theta_0 + \theta_1 x^{(3)} - y^{(3)})\right\} = 0$$

$$\frac{\partial E}{\partial \theta_1} = \frac{2}{3}\left\{(\theta_0 + \theta_1 x^{(1)} - y^{(1)})(x^{(1)}) + (\theta_0 + \theta_1 x^{(2)} - y^{(2)})(x^{(2)}) + (\theta_0 + \theta_1 x^{(3)} - y^{(3)})(x^{(3)})\right\} = 0$$

.

$$E = \frac{1}{3}\left\{(\theta_0 + \theta_1 x^{(1)} - y^{(1)})^2 + (\theta_0 + \theta_1 x^{(2)} - y^{(2)})^2 + (\theta_0 + \theta_1 x^{(3)} - y^{(3)})^2\right\}$$

Finding the gradient

Setting the derivative of E to zero gives two linear equations in two unknowns.

$$3\theta_0 + (x^{(1)} + x^{(2)} + x^{(3)})\theta_1 = (y^{(1)} + y^{(2)} + y^{(3)})$$

$$(x^{(1)} + x^{(2)} + x^{(3)})\theta_0 + (x^{(1)}x^{(1)} + x^{(2)}x^{(2)} + x^{(3)}x^{(3)})\theta_1 = (x^{(1)}y^{(1)} + x^{(2)}y^{(2)} + x^{(3)}y^{(3)})$$

.

$$E = \frac{1}{3}\left\{(\theta_0 + \theta_1 x^{(1)} - y^{(1)})^2 + (\theta_0 + \theta_1 x^{(2)} - y^{(2)})^2 + (\theta_0 + \theta_1 x^{(3)} - y^{(3)})^2\right\}$$

Solving

If we have a linear system we can express it using matrices and vectors.

$$A\mathbf{x} = \mathbf{b}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

We take the dot product of every row in A with x which is equal to the corresponding value in b.

Solving

To solve this equation we find the inverse matrix to A such that when multiplied you get an identity matrix.

$$A^{-1}A = I$$

$$I = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

We multiply by the inverse to solve for **x**.

$$A^{-1}A\mathbf{x} = I\mathbf{x} = \mathbf{x} = A^{-1}\mathbf{b}$$

Solving

Some matrices (degenerate) will not have an inverse. If a matrix is close to degenerate, the inverse may be affected by numerical errors in the computation.

Solving the Linear Regression

Setting the derivative of E to zero gives two linear equations in two unknowns.

$$3\theta_0 + (x^{(1)} + x^{(2)} + x^{(3)})\theta_1 = (y^{(1)} + y^{(2)} + y^{(3)})$$

$$(x^{(1)} + x^{(2)} + x^{(3)})\theta_0 + (x^{(1)}x^{(1)} + x^{(2)}x^{(2)} + x^{(3)}x^{(3)})\theta_1 = (x^{(1)}y^{(1)} + x^{(2)}y^{(2)} + x^{(3)}y^{(3)})$$

$$\begin{bmatrix} 1 & 1 & 1 \\ x^{(1)} & x^{(2)} & x^{(3)} \end{bmatrix} \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ 1 & x^{(3)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x^{(1)} & x^{(2)} & x^{(3)} \end{bmatrix} \begin{bmatrix} y^{(1)} \\ y^{(1)} \\ y^{(1)} \end{bmatrix}$$

$$(\mathbf{X}^T\mathbf{X})\theta = \mathbf{X}^T\mathbf{Y}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ 1 & x^{(3)} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y^{(1)} \\ y^{(1)} \\ y^{(1)} \end{bmatrix}$$

$$\mathbf{X}^T = \begin{bmatrix} 1 & 1 & 1 \\ x^{(1)} & x^{(2)} & x^{(3)} \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

The number of rows in X and Y correspond to the number of training instances (M).

The number of columns in X corresponds to the number of parameters (N)

The number of features used is one less than the number of parameters.

Solving the Linear Regression

$$(\mathbf{X}^T\mathbf{X})\theta = \mathbf{X}^T\mathbf{y}$$

Solving, we have: $\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

The **normal equation** gives a direct solution for the values of our parameters that minimizes the RMSE cost function.

Performance

This performs well with larger training sets and scales linearly with the number of predictions to be made.

The main work in this solution corresponds to working with the N by N matrix $(\mathbf{X}^T\mathbf{X})$

Computing each of the N squared values in this matrix requires time that is O(M) for a total of $O(M\ N^2)$.

Inverting the matrix is between $O(N^{2.5})$ to $O(N^3)$ depending on the technique used.

SKlearn

1) Create a model

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
```

2) Fit the model to data

```
X = train_set[[“feature”]]  #array like
y = train_set[“target”]     #vector like
lin_reg.fit(X,y)
```

3) Score the model

```
r2 = lin_reg.score(X,y)
```

4) Evaluate the model with the test data find the MSE and R2

```
from sklearn.metrics import mean_squared_error
X_eval = test_set[[“feature”]]
y_true = test_set[“target”]
y_pred = lin_reg.predict(X_eval)
r2 = line_reg.score(X_eval, y_true)
rms_error = mean_squared_error(y_true, y_pred)
```

5) Use the model to make predictions with new data.

```
X_predict = [[17], [20], [19]]     #3 instances
y_predict = lin_reg.predict(X_predict)
```

SKlearn

We can fit more features from the data set as desired.

```
X = train_set[[“feature”, “other_feature]]
y = train_set[“target”]
lin_reg.fit(X,y)
```

```
from sklearn.metrics import mean_squared_error
X_eval = test_set[[“feature”, “other_feature]]
y_true = test_set[“target”]

y_pred = lin_reg.predict(X_eval)
r2 = line_reg.score(X_eval, y_true)
rms_error = mean_squared_error(y_true, y_pred)
```

We now need to provide two pieces of data for each instance that we make a predictions for.

```
X_predict =[[17, 9], [20, 7], [19, 4]]
y_predict = lin_reg.predict(X_predict)
```

R2 - Coefficient of Determination

The coefficient of determination tells us what percentage of the variation in our output y is determined by the variation in the input x. The maximum value it can have is 1. It can be arbitrarily bad (negative).

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Residual - Difference between the true y value and the predicted y.

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y})^2$$

Total - Difference between the true y value and the average y

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Root Mean Square Error

Roughly, this is a standard deviation.

For a Gaussian distribution, we expect that

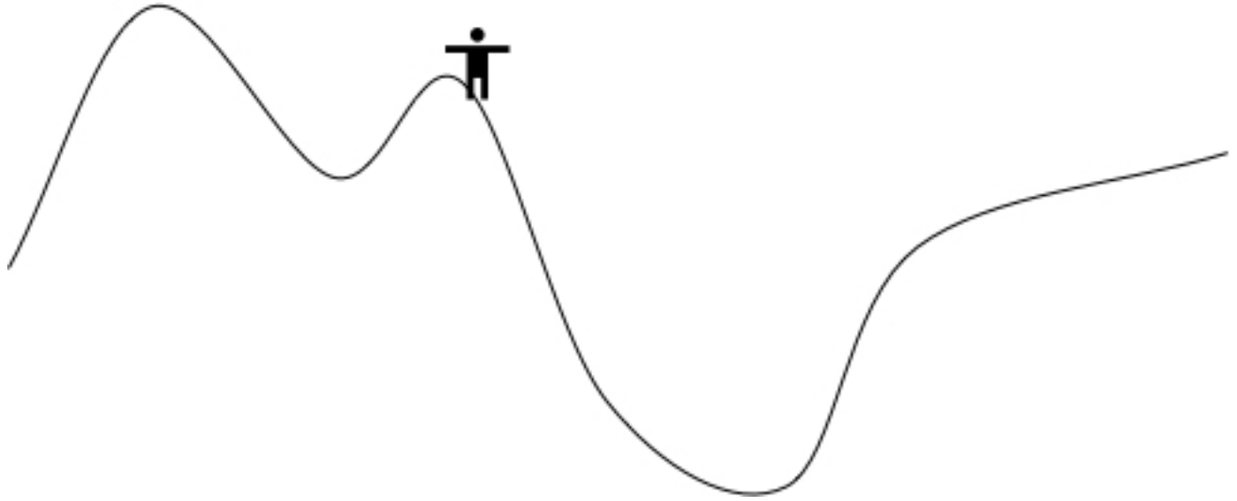
1) 68% of the data is within 1 standard deviations.

2) 95% of the data is within 2 standard deviations.

3) 99.7% of the data is within 2 standard deviations.

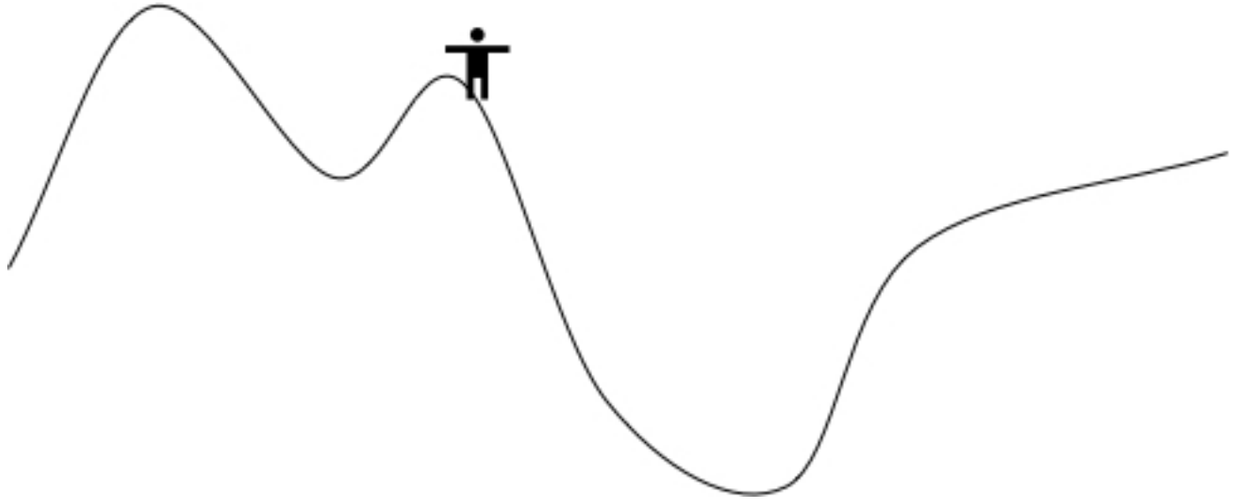
Find minimums (General)

Suppose that we want to get to the bottom of a valley. How can we do it?



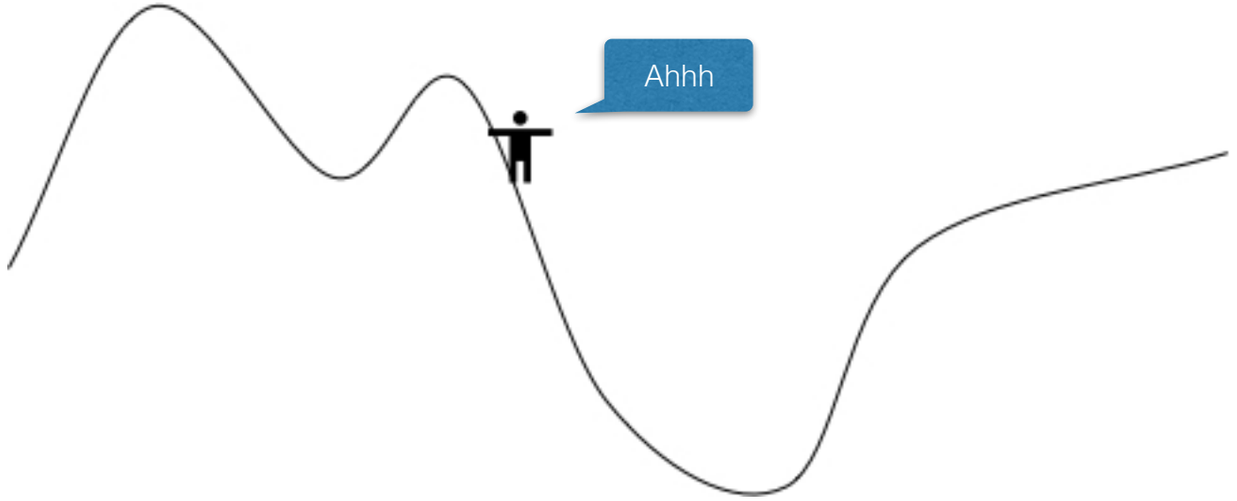
Gradient Descent

Find the direction of maximum change and take a step.



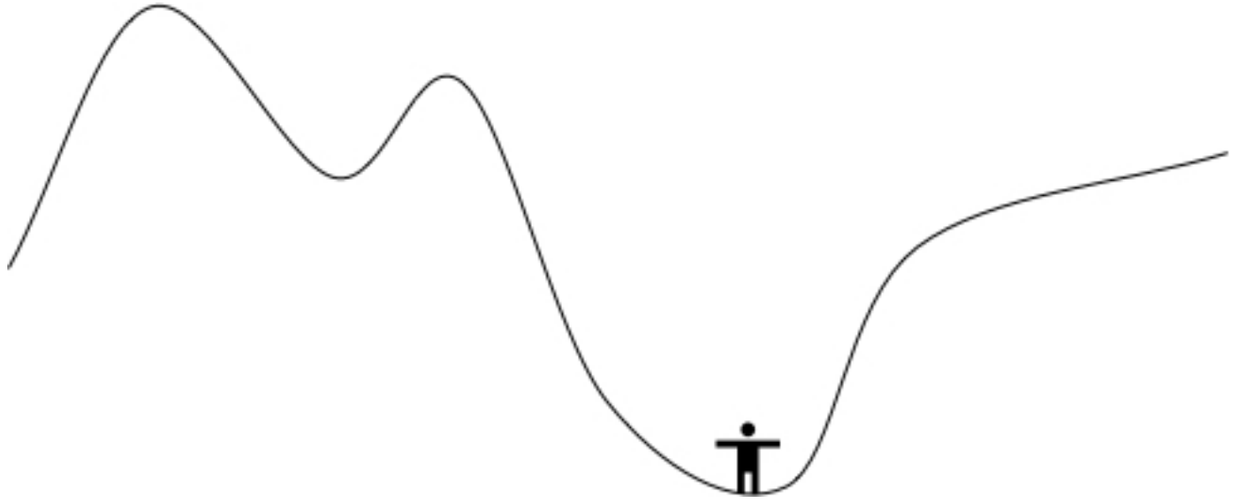
Gradient Descent

Find the direction of maximum change and take a step.



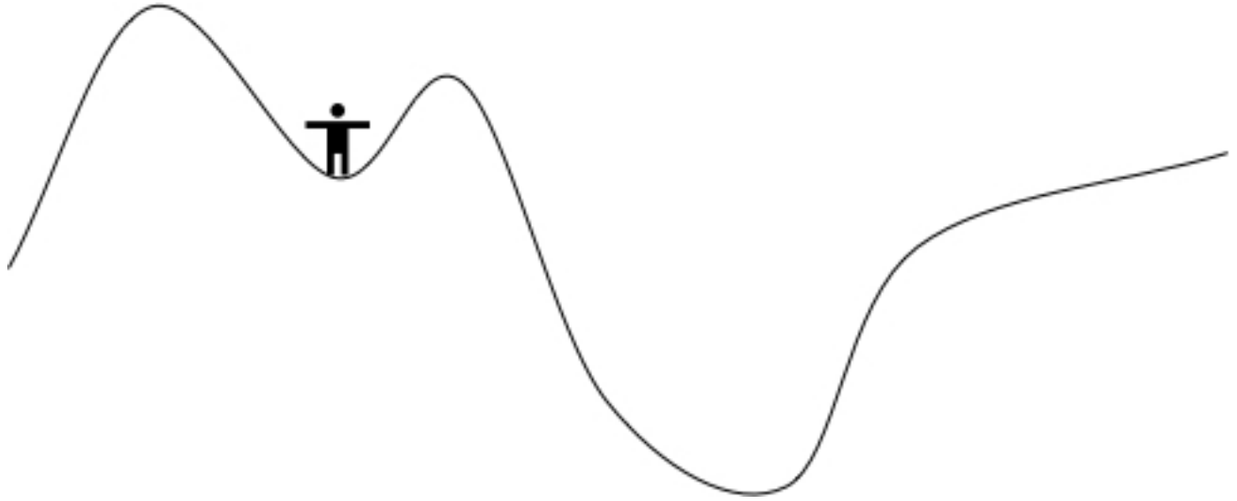
Gradient Descent

Find the direction of maximum change and take a step.



Gradient Descent

When the gradient is zero* we have discovered a critical point. It could be a maximum, minimum or an inflection point. The maximum/minimum may be local or global.



** This assumes that the function is well behaved (continuous).*

Gradient Descent

When we perform a gradient descent, we have to be careful how big a step to take.

- If we take too big a step, we may see behavior where we bounce around a minimum, but never settle.
- If we take too small a step, it may take too long to find the minimum and we are less likely to bounce out of a local minimum.
- If we have a shallow gradient, we may converge very slowly to a minimum.
- **Simulated Annealing** - Start with large steps and gradually decrease the step size over time. You may group steps into epochs and remember the minimum discovered.

Gradient Descent

Suppose that we have two features f1 and f2 where the range of values for f1 is 1:10 and the range of values for f2 is 1:10000. When we do our gradient descent, we can end up in a situation where we quickly minimize f1 and then slowly converge on creature f2. In order to avoid such situations, it is common to **rescale features** so that they all have similar range.

Questions

- What are the parameters for a linear regression with one input feature?
- What are the parameters for a linear regression with 4 input features?
- What is the bias for a linear regression?
- What is the output of a linear regression model?
- What is an L1 norm?
- What is an L2 norm?
- What are MAE, MSE, and RMSE? How is the error for each point measured?
- What is the gradient of a function?
- How do we compute the gradient of an error function?
- How do we know that we found a critical point of our error function?
- How do you solve the vector equation Ax=b?
- Can we solve every equation Ax=b?
- What is the normal equation?
- What is the performance of solving the normal equation if we increase the number of points to be fitted?
- What is the performance of solving the normal equation if we increase the number of parameters?
- What is a local vs. global minimum?
- Does solving the normal equation give a local or global minimum?
- What is gradient descent?
- Does gradient descent find a local or a global minimum?
- What problems can gradient descent have if the step size is small?
- What problems can gradient descent have if the step size is large?
- What is simulated annealing.
- Why should we rescale features before doing a linear regression?

Resources

- Stanford Machine Learning
  - CS229-linalg
  - CS229-notes1
- SciKit Linear Regression