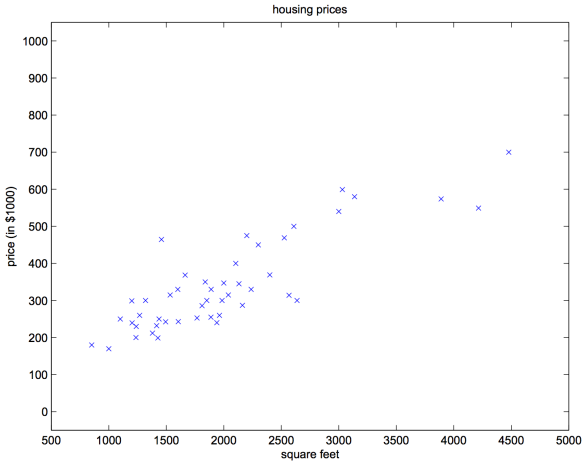
Linear Regression:
Stochastic Gradient Decent

Machine Learning: Linear Regression

Living Area (feet)	Asking Price $ (1000s)
2104	400
1404	232
1534	315
852	178
1940	240
2000	?



m = #training examples
x = input variables/ features
y = output variable
(x,y) = training example
$(x^{(i)},y^{(i)})$ = ith example

Machine Learning: Linear Regression

Living Area (feet)	#bedrooms	Asking Price $ (1000s)
2104	3	400
1404	3	232
1534	3	315
852	2	178
1940	4	240
2000	2	?

```
m     = #training
examples
x     = input variables/
           features
y     = output variable
(x,y) = training example
(x^(i),y^(i)) = ith example
```

Machine Learning: Linear Regression

x_1 = size, x_2 = #bedrooms

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

m = #training examples
x = input variables/ features
y = output variable
(x,y) = training example
$(x^{(i)}, y^{(i)})$ = ith example

Polynomial to predict cost

Theta transpose x is polynomial evaluation

Machine Learning: Linear Regression

x_1 = size, x_2 = #bedrooms

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

We don't know the thetas that will make h(x)=y

There may not be. The relationship may not be linear

Machine Learning: Linear Regression

x_1 = size, x_2 = #bedrooms

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

We don't know the thetas that will make h(x)=y

There may not be. The relationship may not be linear

We can talk about error for some theta.

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

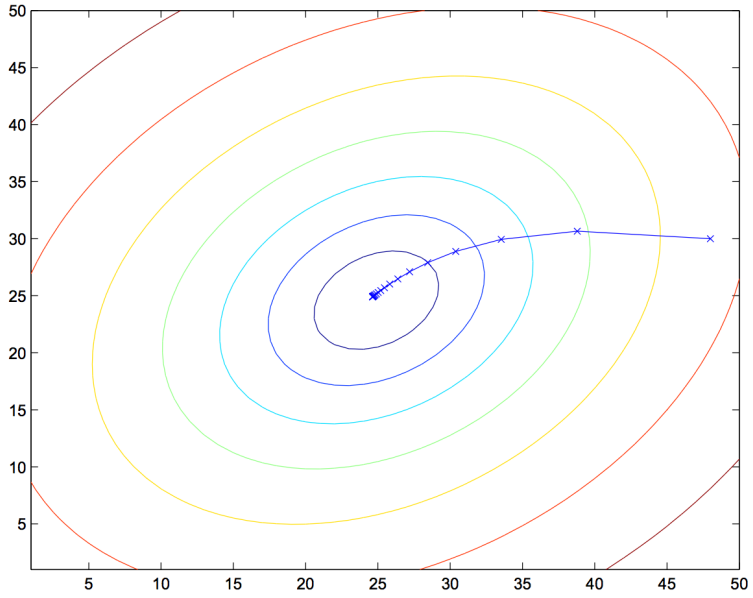
Difference between output and expected for every training example

Machine Learning: Linear Regression

We can talk about how the error will change as we change each theta

$$\frac{\partial}{\partial \theta_j} J(\theta)$$

Machine Learning: Linear Regression: Gradient Descent



We can talk about how the error will change as we change each theta

$$\frac{\partial}{\partial \theta_j} J(\theta)$$

Then if we adjust theta by a small amount to minimize the error our function will have overall lower error

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Machine Learning: Linear Regression: Gradient Descent

What is the change of J with respect to theta j.
Lets look at it at one training example (x,y).

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y\right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y\right) \\
&= \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= \left(h_\theta(x) - y\right) x_j
\end{aligned}
$$

Chain rule. The 2*½ makes it look nice.

Expanding h(x)

constants go away and we are lefts with the x that touches theta j

We can read this as the input variable x_j scaled by the prediction error

Machine Learning: Linear Regression: Gradient Descent

Now we can use this to update our thetas in order to minimize prediction error. Using an alpha term we can control how fast we move through the space.

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

Machine Learning: Linear Regression: Gradient Descent

Now we can use this to update our thetas in order to minimize prediction error. Using an alpha term we can control how fast we move through the space.

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

We can also make a single update to theta by

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

Machine Learning: Linear Regression: Gradient Descent: Sample Code

```
double[] x1 = {2104, 1404, 1534,  852, 1940};
double[] x2 = {   3,    3,    3,    2,    4};
double[] y  = { 400,  232,  315,  178,  240};
double[] th = {   1,    1,    1};
double alpha = 0.00000001;

System.out.println("Initial Error: " + smallNum(error(th, x1, x2,
y)));

for (int i = 0; i < 10; i++)
    updateSGD(th, x1, x2, y, alpha);

System.out.println("Final Error: " + smallNum(error(th, x1, x2, y)));
```

Machine Learning: Linear Regression: Gradient Descent: Sample Code

```
double h(double[] th, double x1, double x2){
  return th[0] + th[1]*x1 + th[2]*x2;
}


double error(double[] th, double[] x1, double[] x2, double[] y){

  double error = 0;
  for (int i = 0; i < x1.length; i++)
    error += Math.pow(h(th,x1[i],x2[i]) - y[i],2);

  return 0.5*error;
}
```

Machine Learning: Linear Regression: Gradient Descent: Sample Code

```
updateGD(double[] th, double[] x1, double[] x2, ...){

  double accth0 = 0;double accth1 = 0;double accth2 = 0;

  for (int i = 0; i < x1.length; i++){
    double instanceError = y[i] - h(th,x1[i],x2[i]);
    accth0 += instanceError * 1;
    accth1 += instanceError * x1[i];
    accth2 += instanceError * x2[i];
  }

  th[0] += alpha*accth0; th[1] += alpha*accth1; th[2] += alpha*accth2;

}
```

Machine Learning: Linear Regression: Gradient Descent: Sample Code

```
updateSGD(double[] th, double[] x1, double[] x2, ...){

  for (int i = 0; i < x1.length; i++){

    double instanceError = y[i] - h(th,x1[i],x2[i]);

    th[0] += alpha*instanceError * 1;
    th[1] += alpha*instanceError * x1[i];
    th[2] += alpha*instanceError * x2[i];
  }
}
```

Machine Learning: Linear Regression: Gradient Descent

```
Given: x1=2104.0,x2=3.0, pred=2108.00, exp=400.0
Given: x1=1404.0,x2=3.0, pred=1408.00, exp=232.0
Given: x1=1534.0,x2=3.0, pred=1538.00, exp=315.0
Given: x1=852.0,x2=2.0, pred=855.00, exp=178.0
Given: x1=1940.0,x2=4.0, pred=1945.00, exp=240.0
Initial Error: 4580661.50
Update thetas (GD): [1.0, 1.0, 1.0]->[0.999, 0.889, 0.999]
Given: x1=2104.0,x2=3.0, pred=1876.45, exp=400.0
Given: x1=1404.0,x2=3.0, pred=1253.48, exp=232.0
Given: x1=1534.0,x2=3.0, pred=1369.18, exp=315.0
Given: x1=852.0,x2=2.0, pred=761.23, exp=178.0
Given: x1=1940.0,x2=4.0, pred=1731.50, exp=240.0
Final Error: 3449670.08
```

Machine Learning: Linear Regression: Gradient Descent

```
Given: x1=2104.0,x2=3.0, pred=2108.00, exp=400.0
Given: x1=1404.0,x2=3.0, pred=1408.00, exp=232.0
Given: x1=1534.0,x2=3.0, pred=1538.00, exp=315.0
Given: x1=852.0,x2=2.0, pred=855.00, exp=178.0
Given: x1=1940.0,x2=4.0, pred=1945.00, exp=240.0
Initial Error: 4580661.50
Update thetas (GD): [1.0, 1.0, 1.0]->[0.999, 0.889, 0.999]
... 98 iterations
Update thetas (GD): [0.999, 0.168, 0.998]->[0.999, 0.168, 0.998]
Given: x1=2104.0,x2=3.0, pred=359.21, exp=400.0
Given: x1=1404.0,x2=3.0, pred=241.03, exp=232.0
Given: x1=1534.0,x2=3.0, pred=262.98, exp=315.0
Given: x1=852.0,x2=2.0, pred=146.84, exp=178.0
Given: x1=1940.0,x2=4.0, pred=332.52, exp=240.0
Final Error: 6991.39
```

Machine Learning: Linear Regression: Stochastic Gradient Descent

```
Given: x1=2104.0,x2=3.0, pred=2108.00, exp=400.0
Given: x1=1404.0,x2=3.0, pred=1408.00, exp=232.0
Given: x1=1534.0,x2=3.0, pred=1538.00, exp=315.0
Given: x1=852.0,x2=2.0, pred=855.00, exp=178.0
Given: x1=1940.0,x2=4.0, pred=1945.00, exp=240.0
Initial Error: 4580661.50
Update thetas (SGD): [1.0, 1.0, 1.0]->[0.99, 0.89, 0.99]
Given: x1=2104.0,x2=3.0, pred=1887.46, exp=400.0
Given: x1=1404.0,x2=3.0, pred=1260.83, exp=232.0
Given: x1=1534.0,x2=3.0, pred=1377.21, exp=315.0
Given: x1=852.0,x2=2.0, pred=765.69, exp=178.0
Given: x1=1940.0,x2=4.0, pred=1741.65, exp=240.0
Final Error: 3499831.32
```

Machine Learning: Linear Regression: Stochastic Gradient Descent

```
Given: x1=2104.0,x2=3.0, pred=2108.00, exp=400.0
Given: x1=1404.0,x2=3.0, pred=1408.00, exp=232.0
Given: x1=1534.0,x2=3.0, pred=1538.00, exp=315.0
Given: x1=852.0,x2=2.0, pred=855.00, exp=178.0
Given: x1=1940.0,x2=4.0, pred=1945.00, exp=240.0
Initial Error: 4580661.50
Update thetas (SGD): [1.0, 1.0, 1.0]->[0.99, 0.89, 0.99]
... 98 iterations
Update thetas (SGD): [0.99, 0.16, 0.99]->[0.99, 0.16, 0.99]
Given: x1=2104.0,x2=3.0, pred=357.37, exp=400.0
Given: x1=1404.0,x2=3.0, pred=239.80, exp=232.0
Given: x1=1534.0,x2=3.0, pred=261.64, exp=315.0
Given: x1=852.0,x2=2.0, pred=146.09, exp=178.0
Given: x1=1940.0,x2=4.0, pred=330.82, exp=240.0
Final Error: 6996.47
```

References:

Andrew Ng Machine Learning Course
http://cs229.stanford.edu/notes/cs229-notes1.pdf