Dealing with BIGish Data

Stochastic Gradient Descent

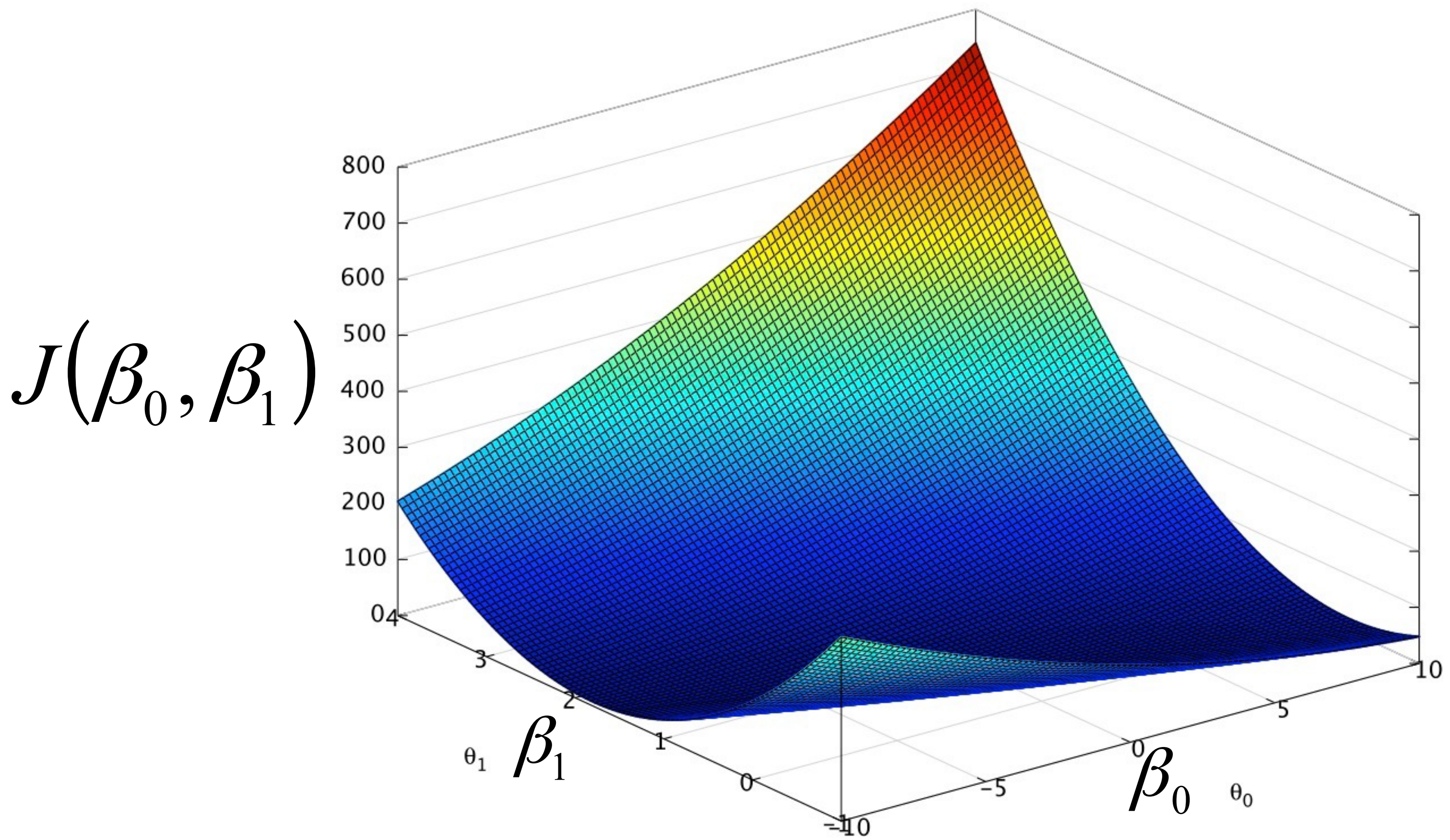
METIS

datascope

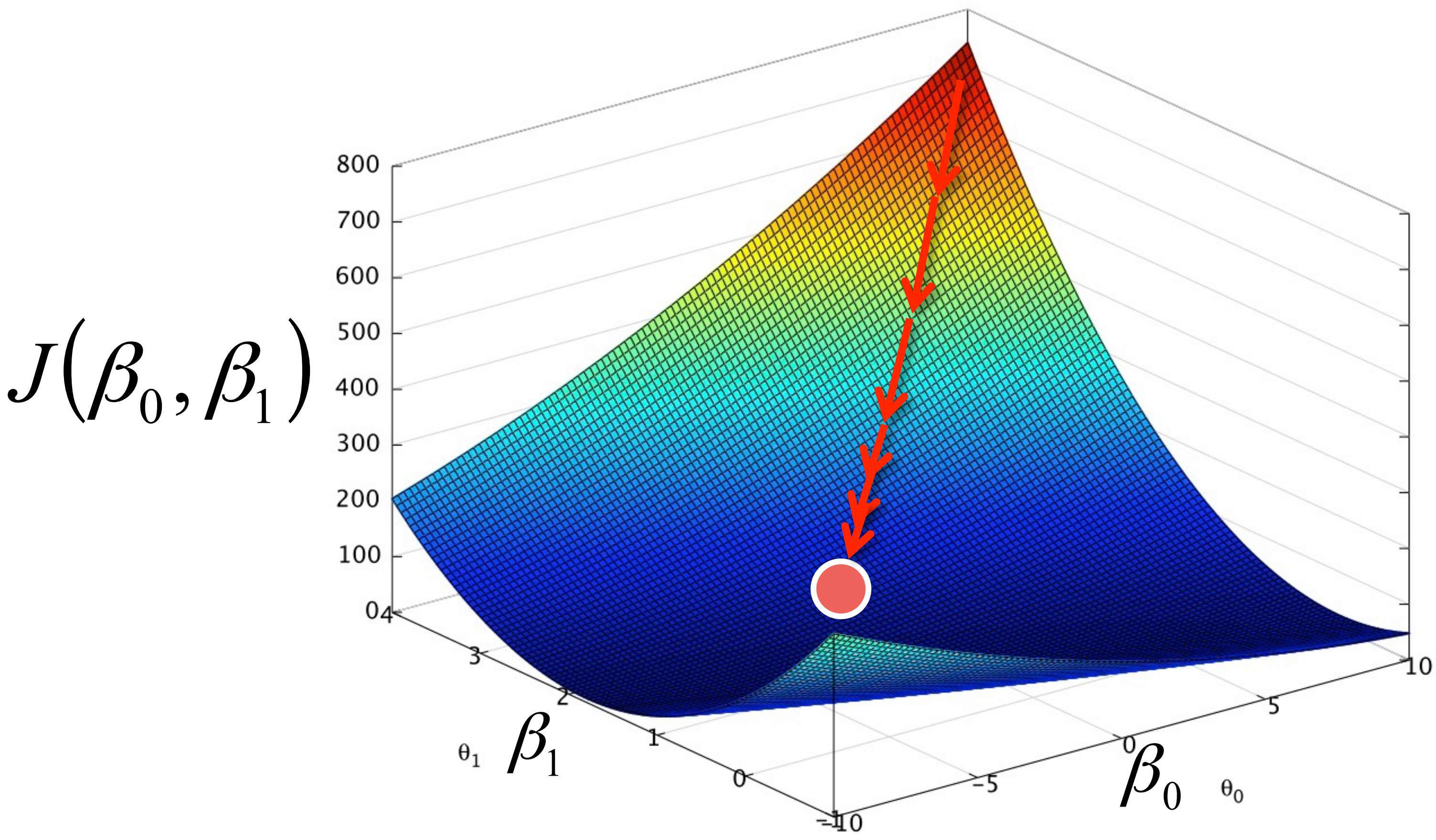
Omg too many points! What do?

Can’t fit memory. Gotta fit a model row by row

Gradient Descent

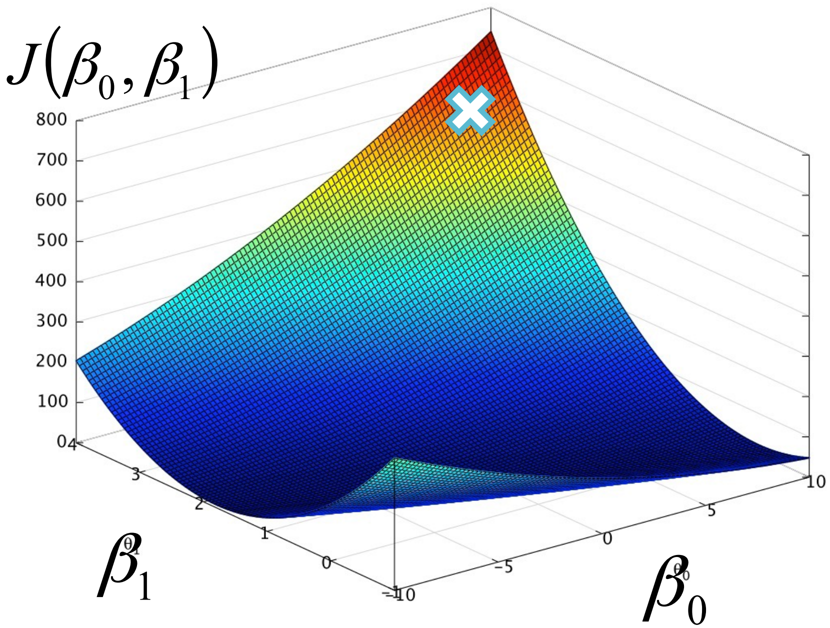


Gradient Descent



Gradient Descent with Linear Regression

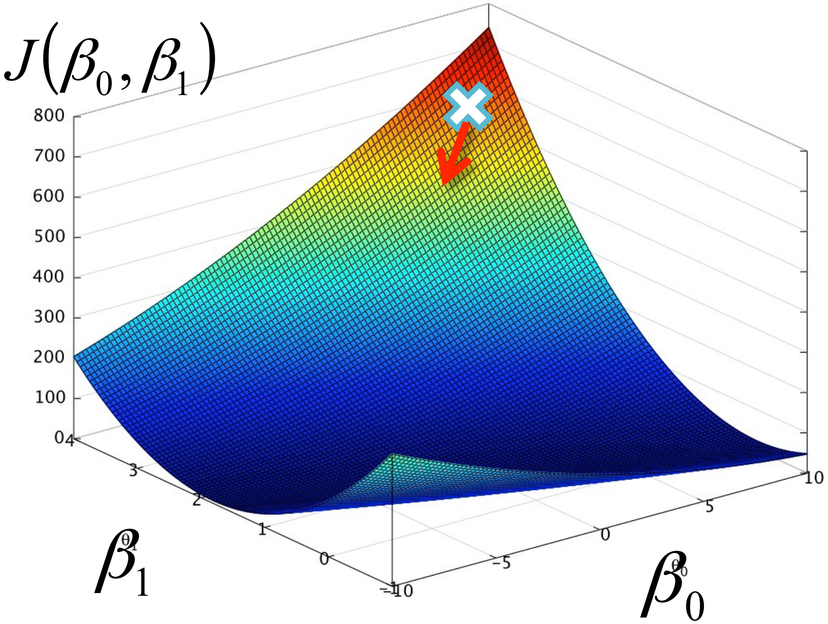
$$J\left(\beta_0,\beta_1\right)=\sum_{i=1}^{m}\left(\left(\beta_0+\beta_1 x_{obs}^{(i)}\right)-y_{obs}^{(i)}\right)^2$$



Gradient Descent with Linear Regression

$$J\left(\beta_0,\beta_1\right)=\sum_{i=1}^{m}\left((\beta_0+\beta_1 x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2$$

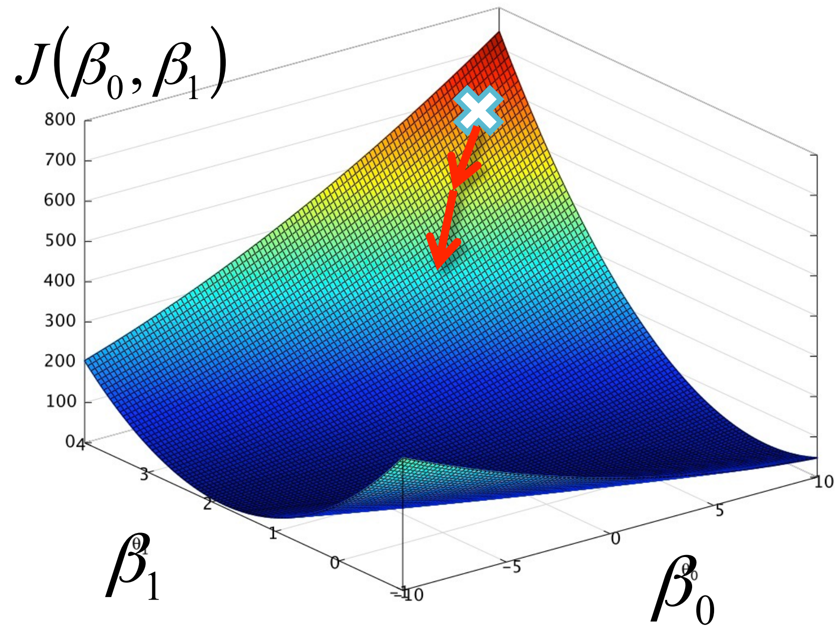
$$w_1=w_0-\alpha\frac{\partial}{\partial\beta_k}\sum_{i=1}^{m}\left((\beta_0+\beta_1 x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2$$



Gradient Descent with Linear Regression

$$J\left(\beta_0,\beta_1\right) = \sum_{i=1}^{m}\left((\beta_0 + \beta_1 x_{obs}^{(i)}) - y_{obs}^{(i)}\right)^2$$

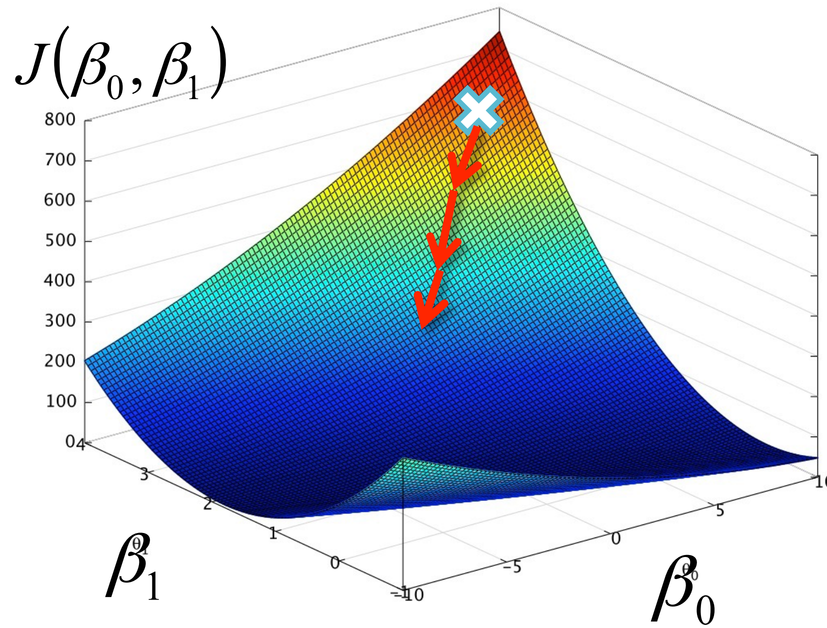
$$w_2 = w_1 - \alpha \frac{\partial}{\partial \beta_k} \sum_{i=1}^{m}\left((\beta_0 + \beta_1 x_{obs}^{(i)}) - y_{obs}^{(i)}\right)^2$$



Gradient Descent with Linear Regression

$$J(\beta_0, \beta_1) = \sum_{i=1}^{m} \left((\beta_0 + \beta_1 x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2$$

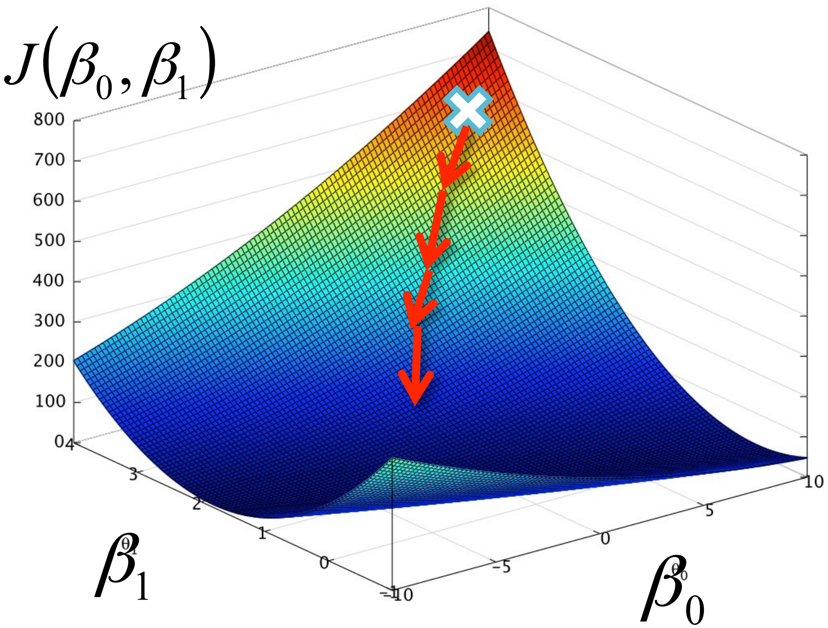
$$w_3 = w_2 - \alpha \frac{\partial}{\partial \beta_k} \sum_{i=1}^{m} \left((\beta_0 + \beta_1 x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2$$



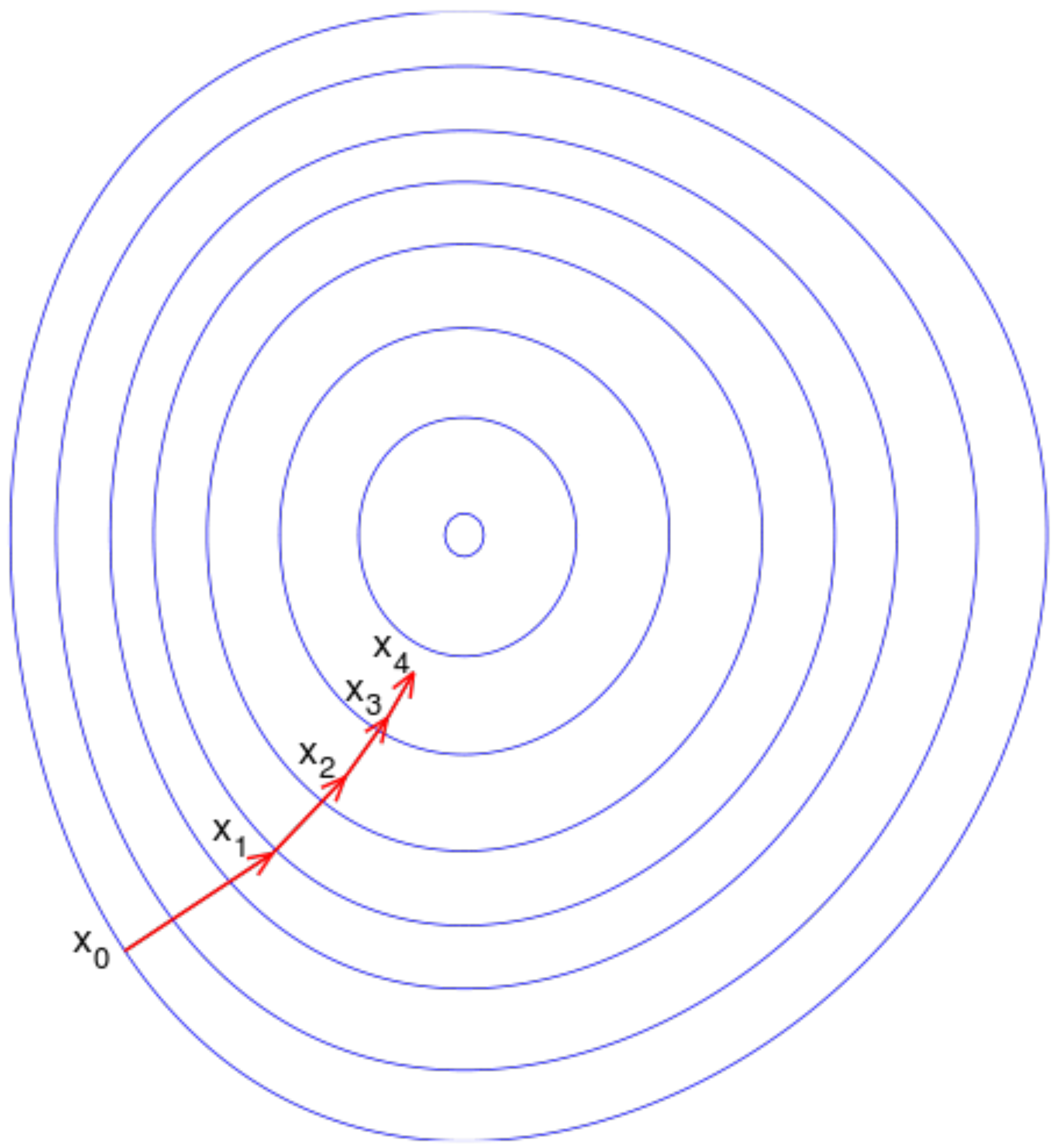
Gradient Descent with Linear Regression

$$J\left(\beta_0,\beta_1\right)=\sum_{i=1}^{m}\left(\left(\beta_0+\beta_1 x_{obs}^{(i)}\right)-y_{obs}^{(i)}\right)^2$$

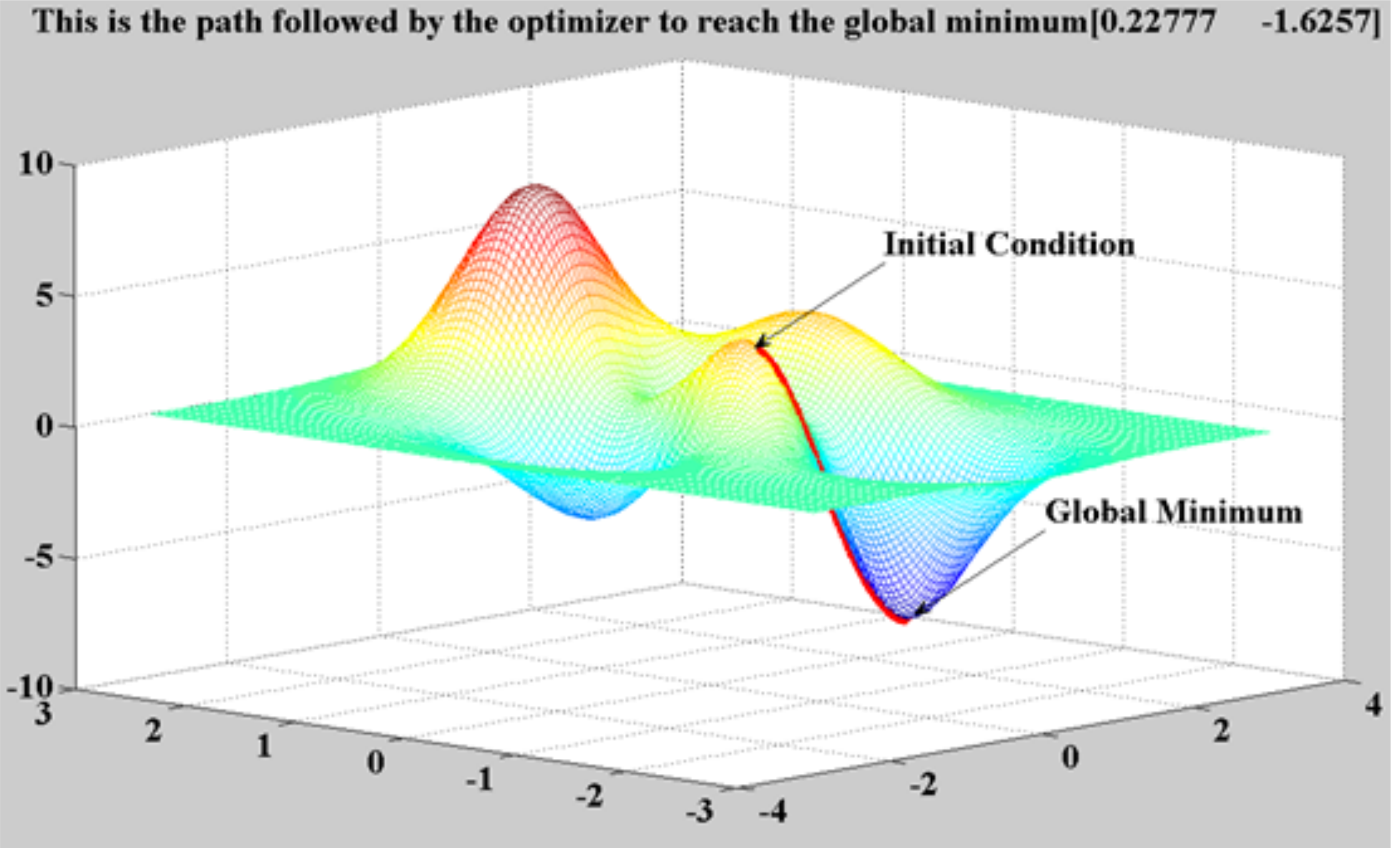
$$w_4=w_3-\alpha\frac{\partial}{\partial\beta_k}\sum_{i=1}^{m}\left(\left(\beta_0+\beta_1 x_{obs}^{(i)}\right)-y_{obs}^{(i)}\right)^2$$



Gradient Descent



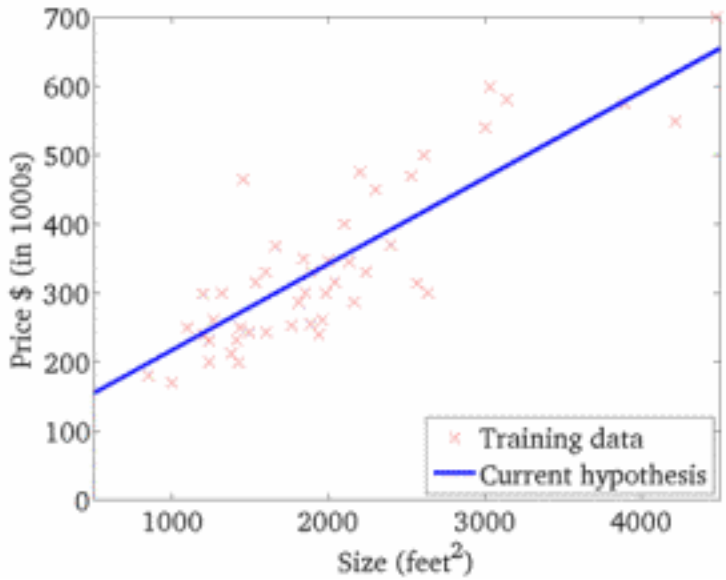
Gradient Descent



Gradient Descent

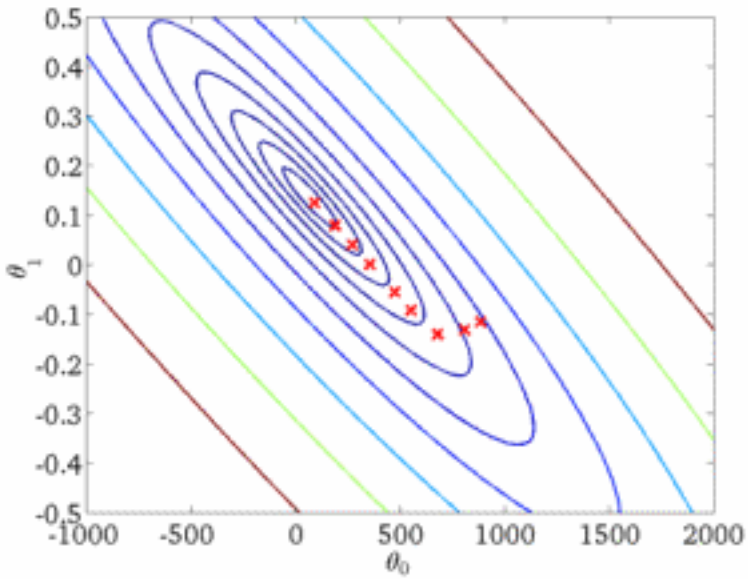
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)

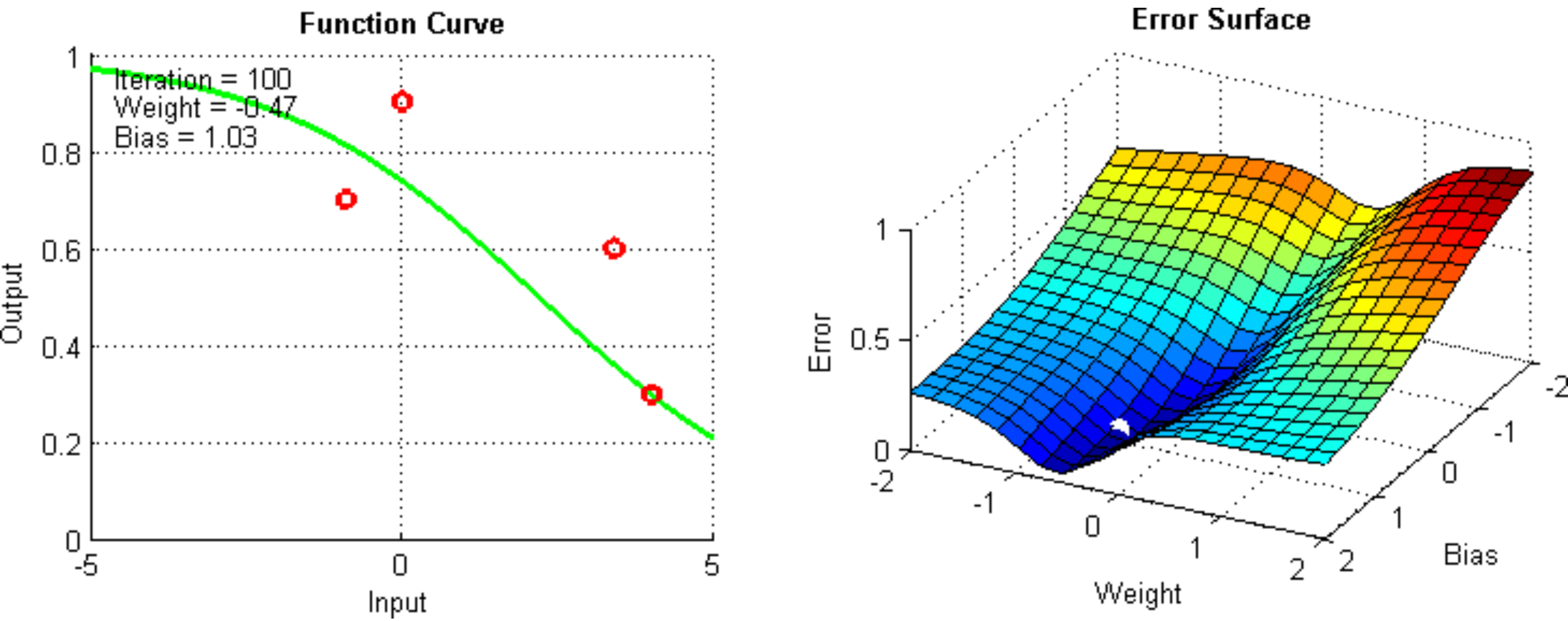


$J(\theta_0, \theta_1)$

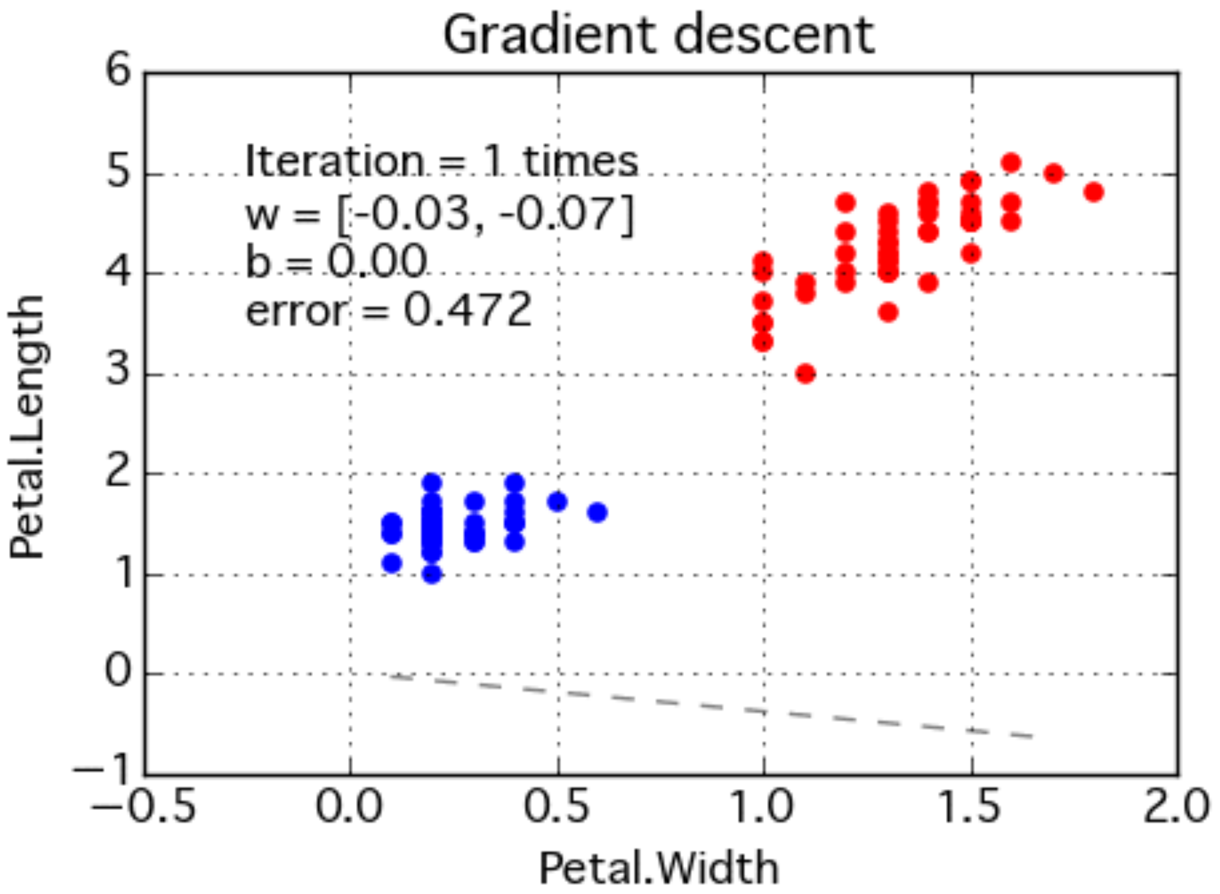
(function of the parameters θ_0, θ_1)



Gradient Descent

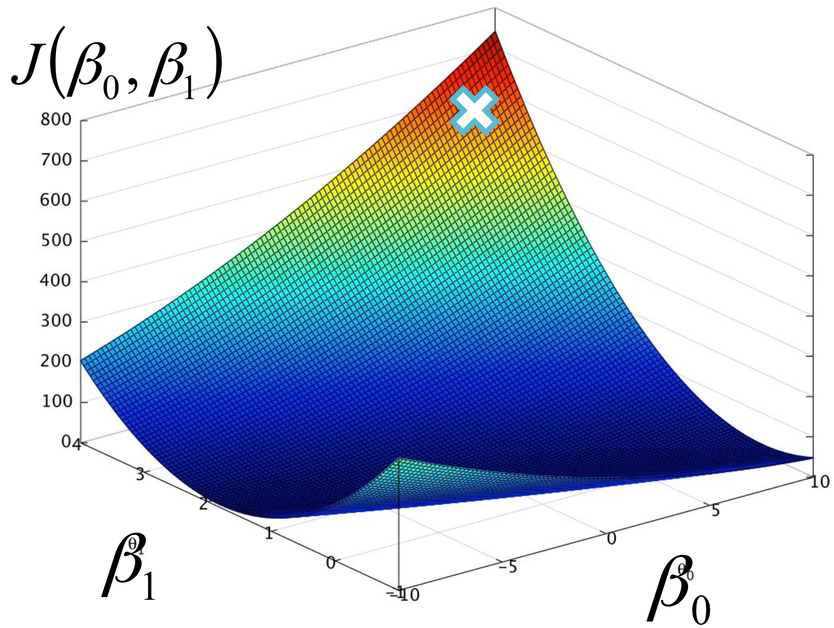


Gradient Descent



Stochastic Gradient Descent

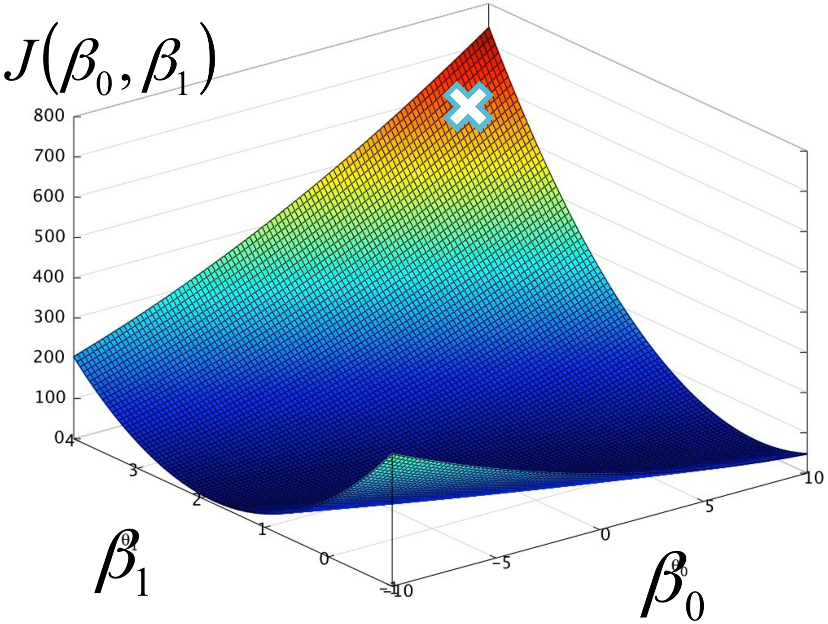
$$J\left(\beta_0,\beta_1\right)=\sum_{i=1}^{m}\left((\beta_0+\beta_1 x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2$$



Stochastic Gradient Descent

$$J(\beta_0, \beta_1) = \sum_{i=1}^{m} \left((\beta_0 + \beta_1 x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2$$

Instead of using all points to find the gradient,
Only use a SINGLE point each time

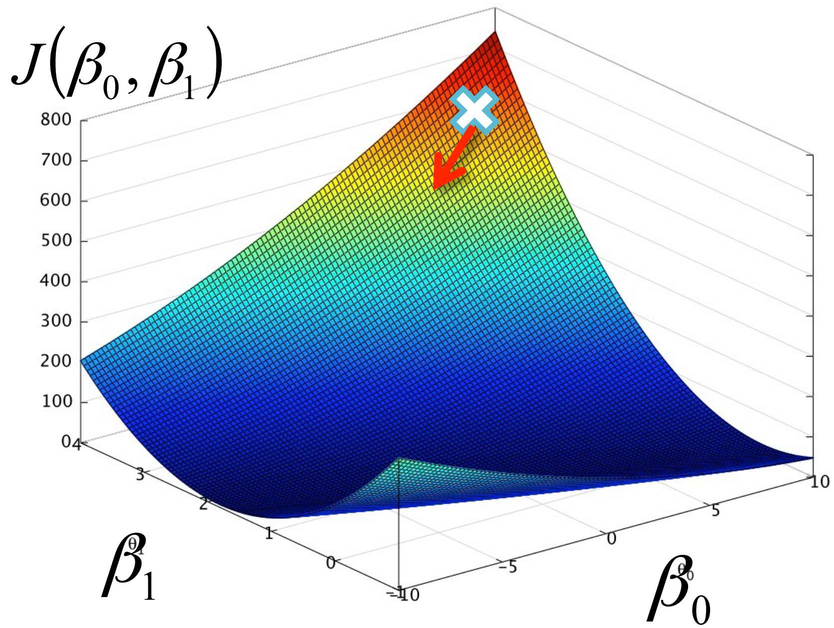


Stochastic Gradient Descent

$$J(\beta_0,\beta_1)=\sum_{i=1}^{m}\left((\beta_0+\beta_1 x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2$$

Instead of using all points to find the gradient,
Only use a SINGLE point each time

$$w_1 = w_0 - \alpha\frac{\partial}{\partial\beta_k}\left((\beta_0+\beta_1 x_{obs}^{(0)})-y_{obs}^{(0)}\right)^2$$

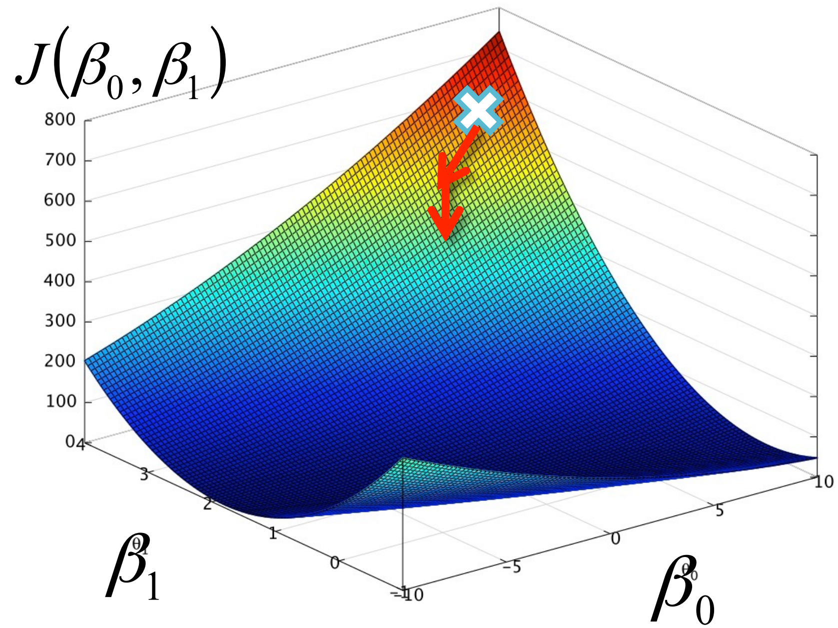


Stochastic Gradient Descent

$$J(\beta_0, \beta_1) = \sum_{i=1}^{m}\left((\beta_0 + \beta_1 x_{obs}^{(i)}) - y_{obs}^{(i)}\right)^2$$

Instead of using all points to find the gradient,
Only use a SINGLE point each time

$$w_2 = w_1 - \alpha \frac{\partial}{\partial \beta_k}\left((\beta_0 + \beta_1 x_{obs}^{(1)}) - y_{obs}^{(1)}\right)^2$$

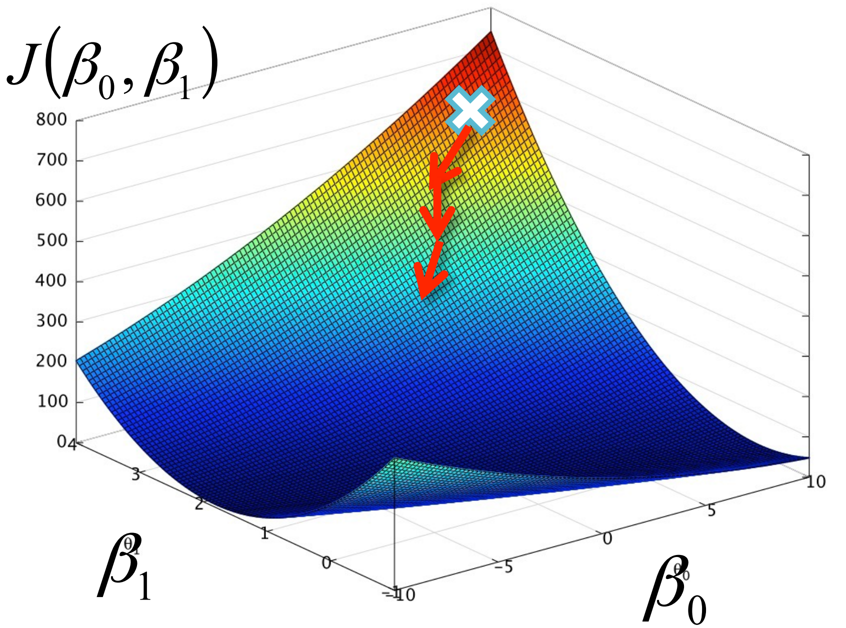


Stochastic Gradient Descent

$$J\left(\beta_0,\beta_1\right)=\sum_{i=1}^{m}\left((\beta_0+\beta_1 x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2$$

Instead of using all points to find the gradient,
Only use a SINGLE point each time

$$w_3=w_2-\alpha\frac{\partial}{\partial\beta_k}\left((\beta_0+\beta_1 x_{obs}^{(2)})-y_{obs}^{(2)}\right)^2$$

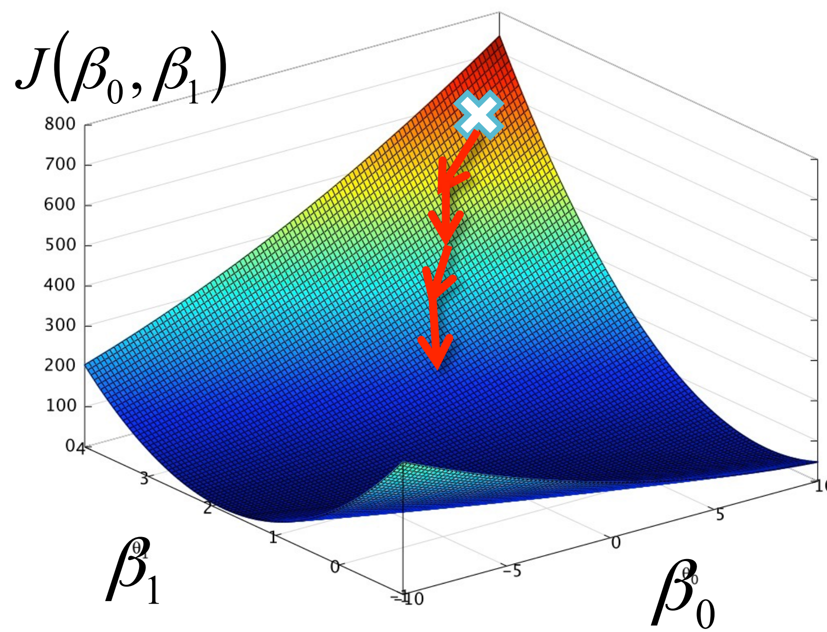


Stochastic Gradient Descent

$$J\left(\beta_0,\beta_1\right)=\sum_{i=1}^{m}\left(\left(\beta_0+\beta_1 x_{obs}^{(i)}\right)-y_{obs}^{(i)}\right)^2$$

Instead of using all points to find the gradient,
Only use a SINGLE point each time

$$w_4=w_3-\alpha\frac{\partial}{\partial\beta_k}\left(\left(\beta_0+\beta_1 x_{obs}^{(3)}\right)-y_{obs}^{(3)}\right)^2$$



Faster

Derivative of single point at each step (instead of 100K)

Online Training

Only need to keep single point in memory

No need to store 100K rows, large data no problem

Covers Many Algorithms

Gradient Descent is the bottleneck for linear algorithms

Can do Linear Regression, Logistic Regression, SVMs

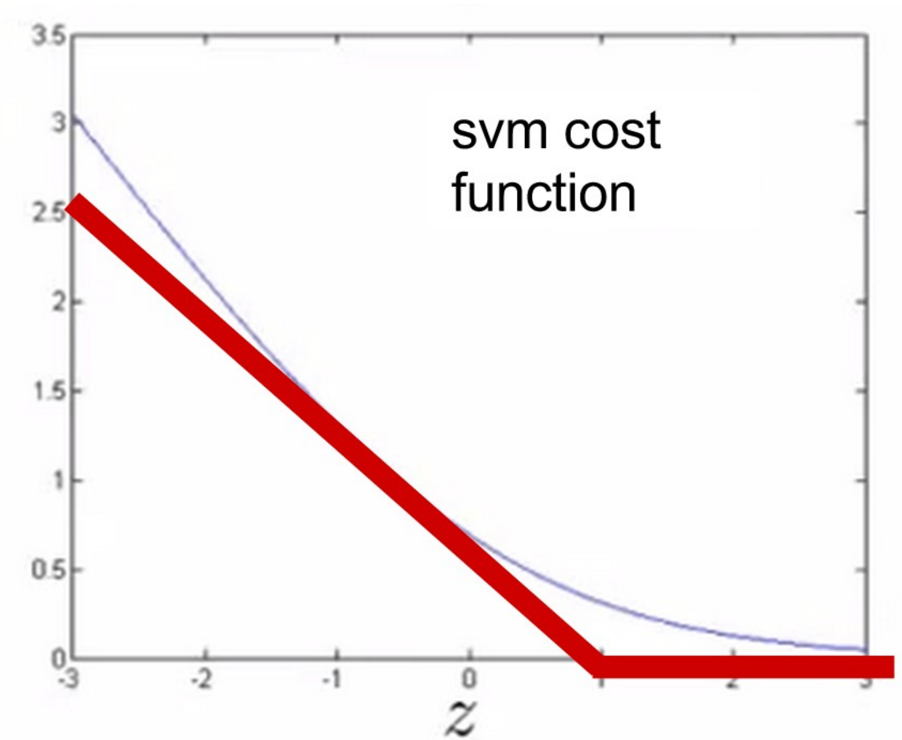
```
from sklearn.linear_model import SGDClassifier
```

```
from sklearn.linear_model import SGDClassifier

SGDClassifier(loss='hinge')
```

from sklearn.linear_model **import** SGDClassifier

SGDClassifier(loss='hinge')

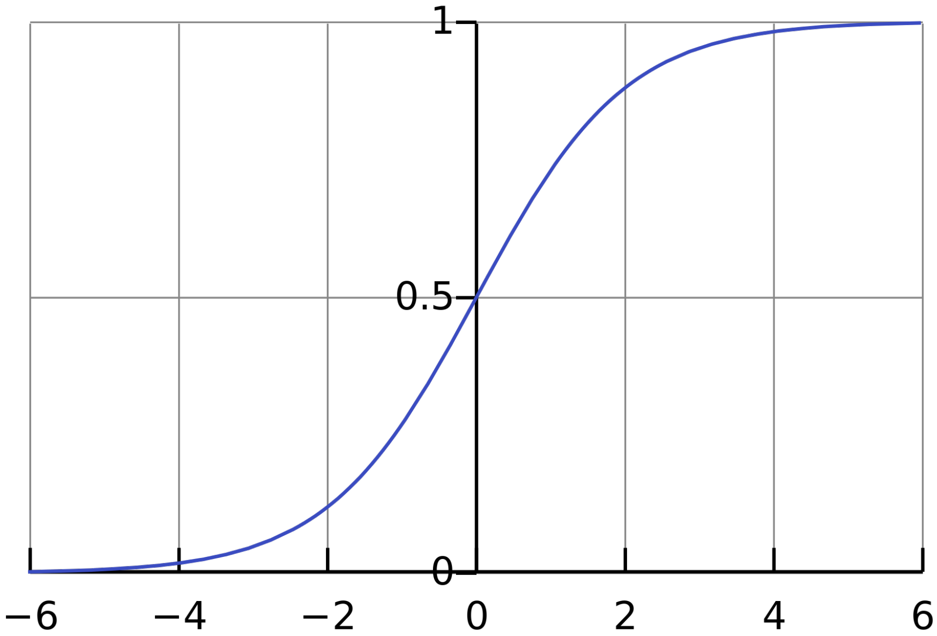


Looks like a hinge.

hinge loss == SVM

```
from sklearn.linear_model import SGDClassifier
```

```
SGDClassifier(loss='log')
```

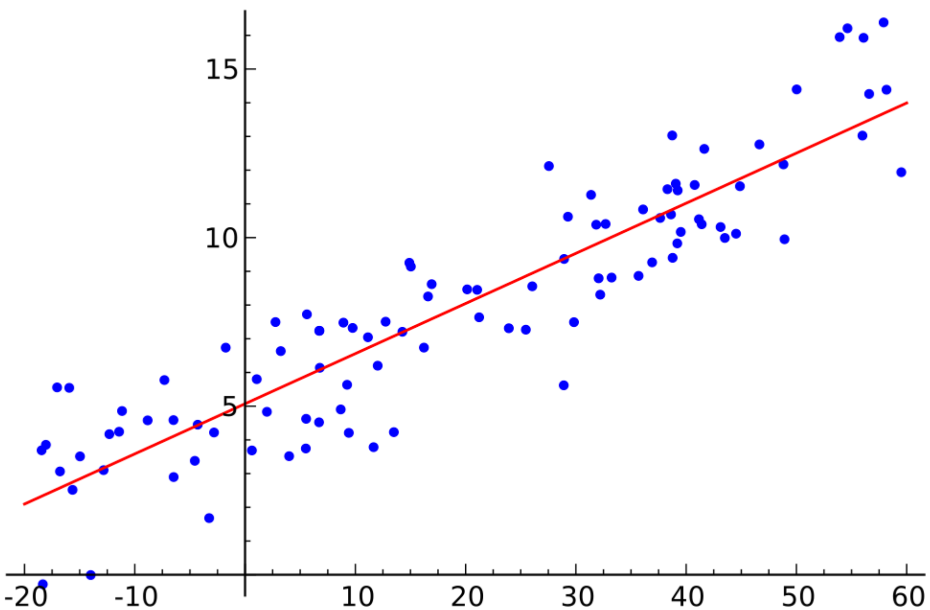


This one's kind of clear

log loss == Logistic Regression

from sklearn.linear_model import SGDRegressor

SGDRegressor(loss='squared_loss')

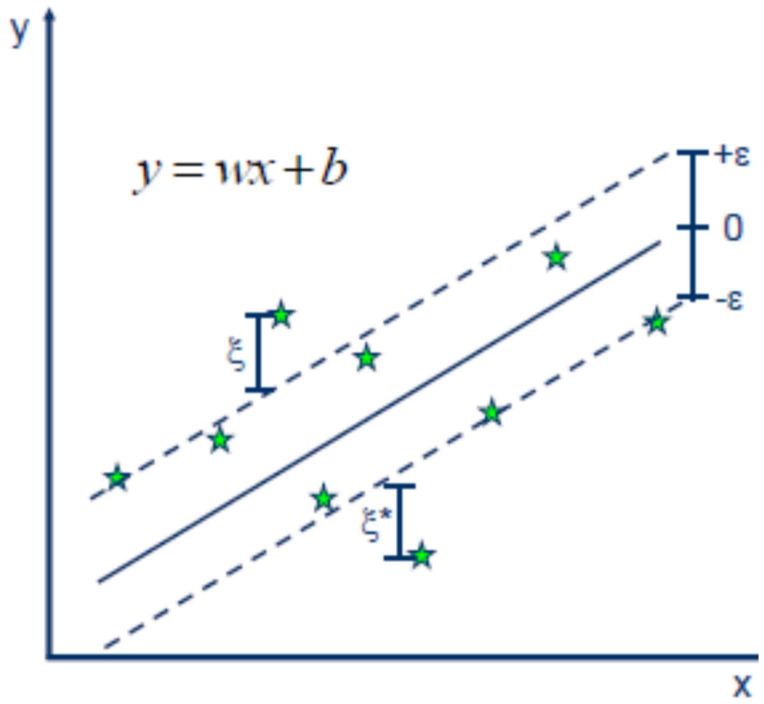


Sum of squared errors

squared loss ==
Linear Regression

```
from sklearn.linear_model import SGDRegressor
```

```
SGDRegressor(loss='epsilon_insensitive')
```



Best loss name ever

epsilon insensitive loss == SVM Regression

```
from sklearn.linear_model import SGDClassifier
```

```
SGDClassifier(alpha=0.0001,
              penalty='l2',
              l1_ratio=0.15)
```

Regularization parameters

Penalty values: 'l1', 'l2', 'elasticnet'