WHERE WE ARE

Supervised Learning

Rules/Trees

Overfitting

Ensembles

Ex: Random Forests

k Nearest Neighbors

Next

- *Optimization with Gradient Descent*
- *Quick Intuition for Logistic Regression*
- *Quick Intuition for Support Vector Machines*
- *Regularization*
- *Stochastic Gradient Descent*

LEARNING = THREE CORE COMPONENTS

Representation

Evaluation

Optimization

- Gradient Descent

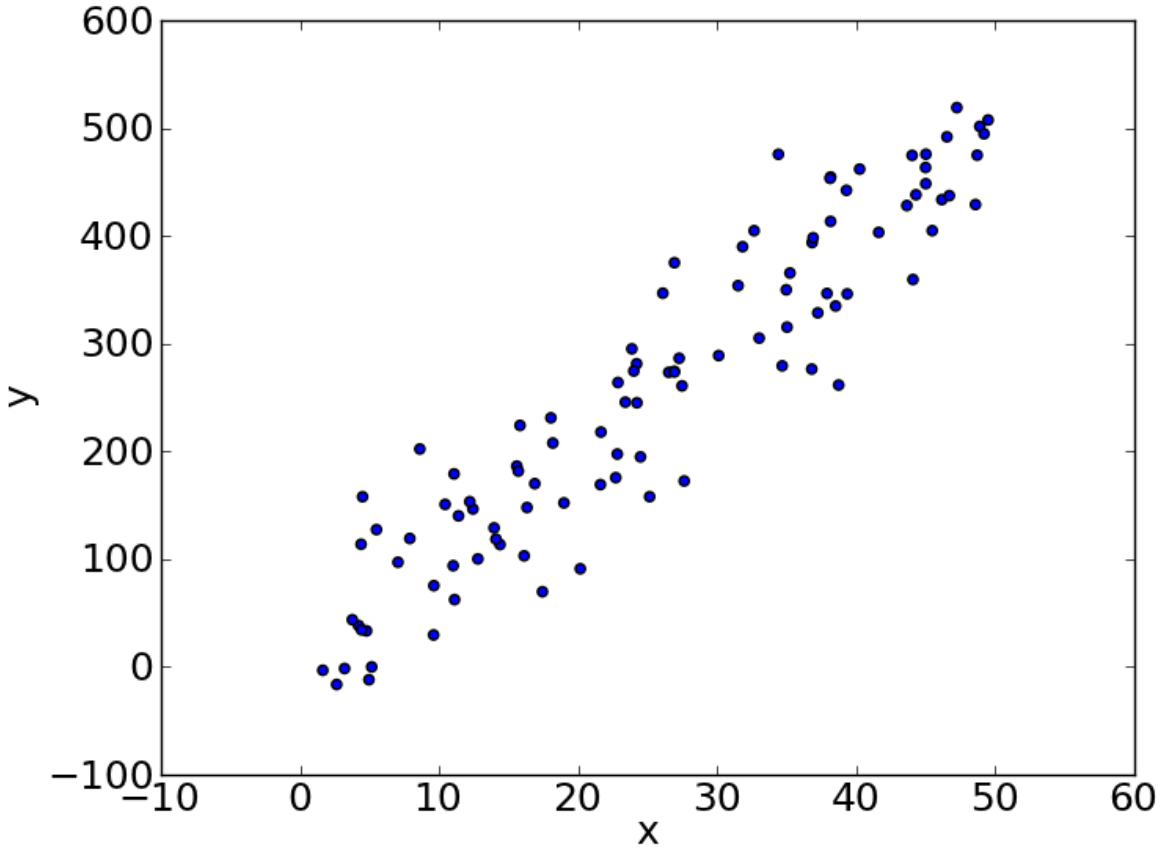
Pedro Domingos, A Few Useful Things to Know about Machine Learning, CACM 55(10), 2012

GRADIENT DESCENT IN A NUTSHELL

Express your learning problem in terms of a cost function that should be minimized

Starting at initial point, step "downhill" until you reach a minimum

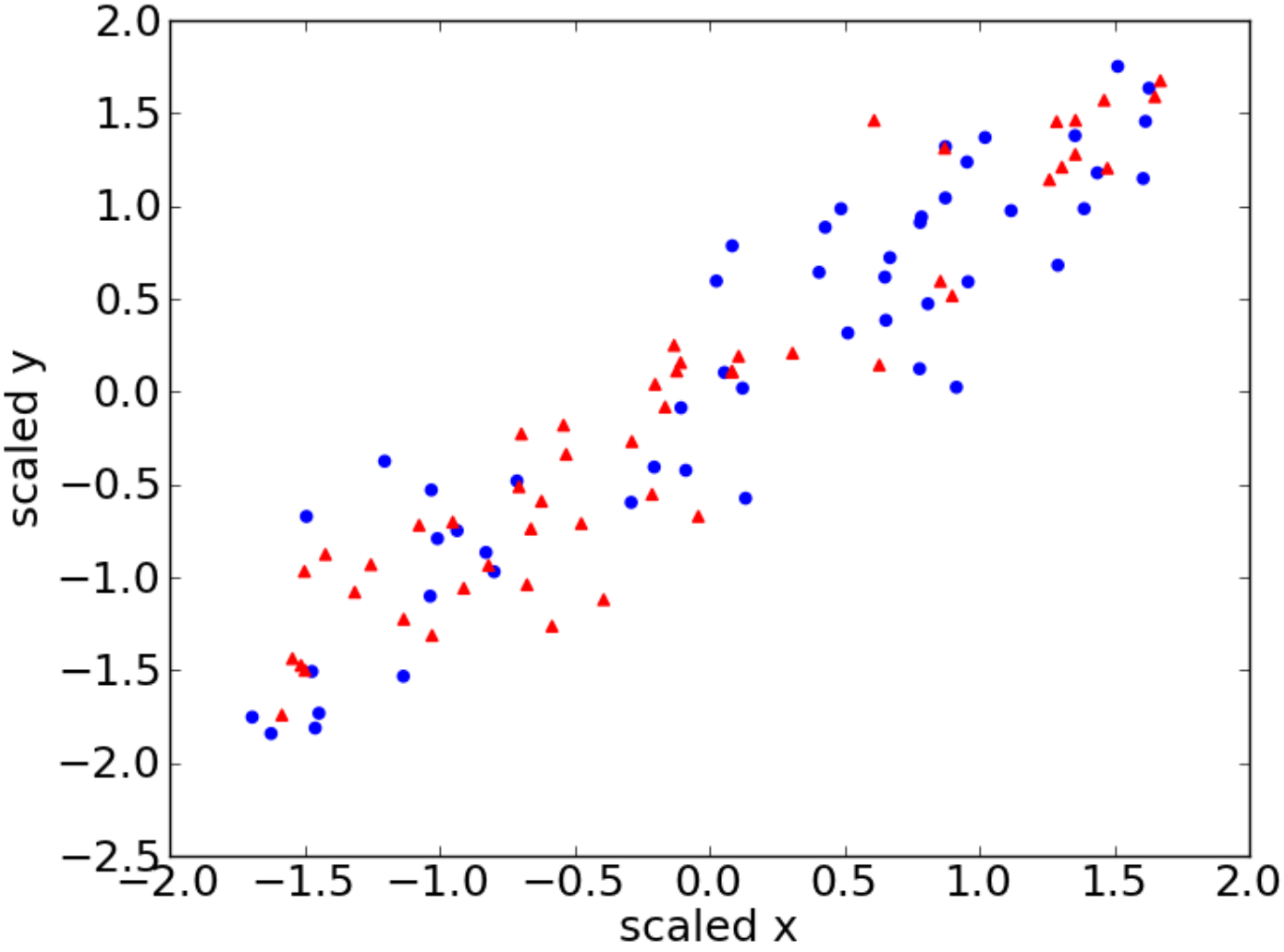
Some situations offer a guarantee that the minimum is the global minimum; others don't

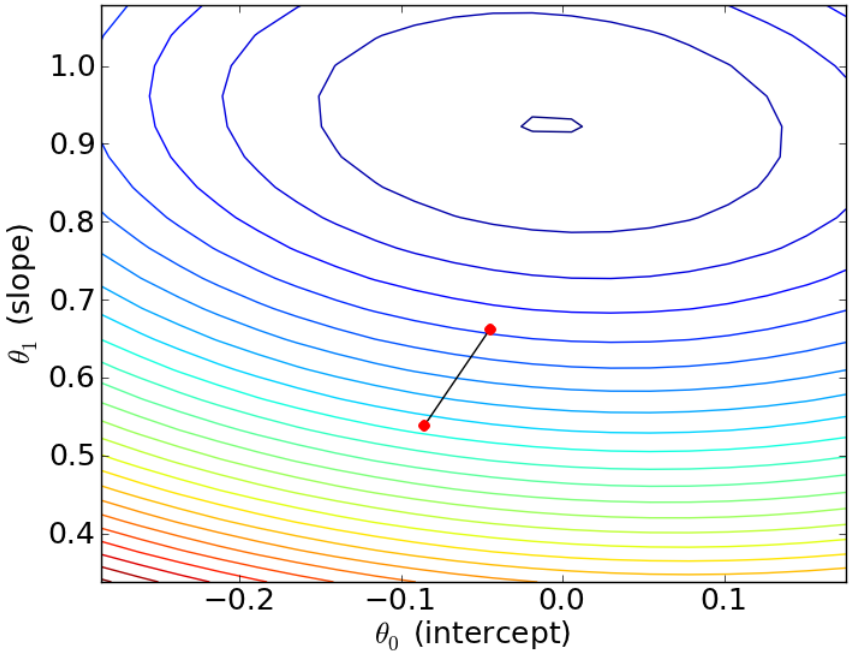


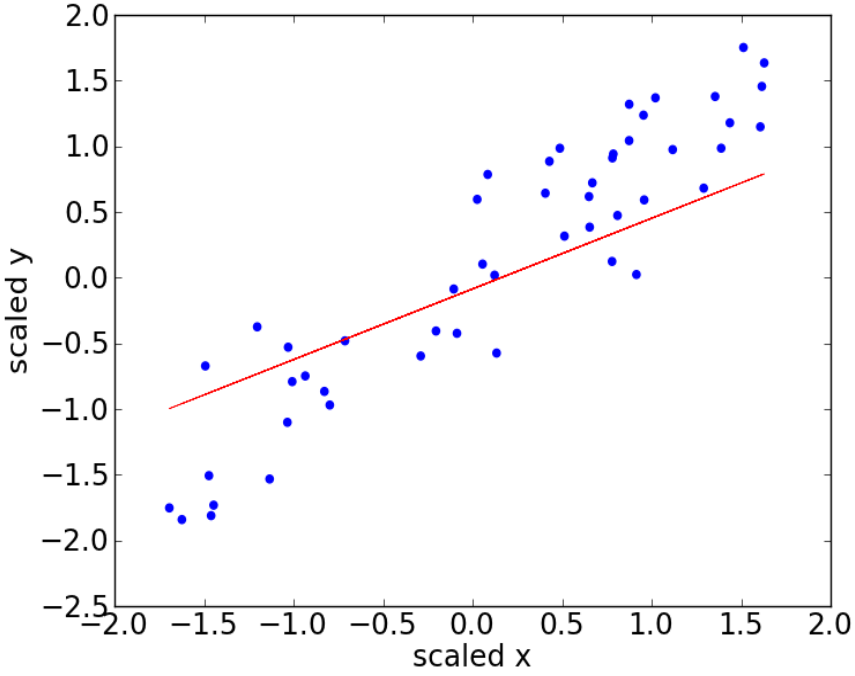
input variable

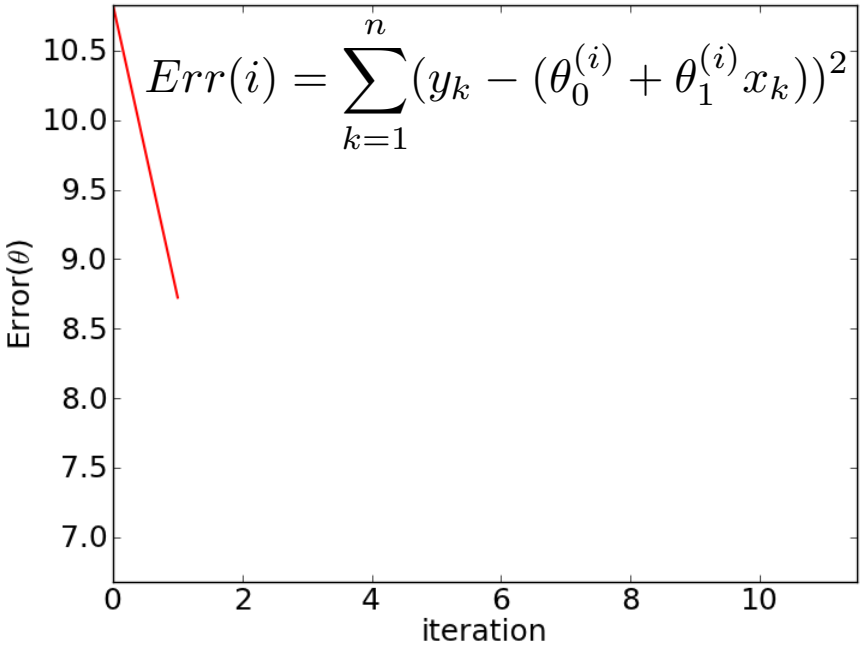
response variable

x	y
3.1	84.2
19.6	175.8
45.9	448.3
6.8	50.4
3.5	81.9
…	…

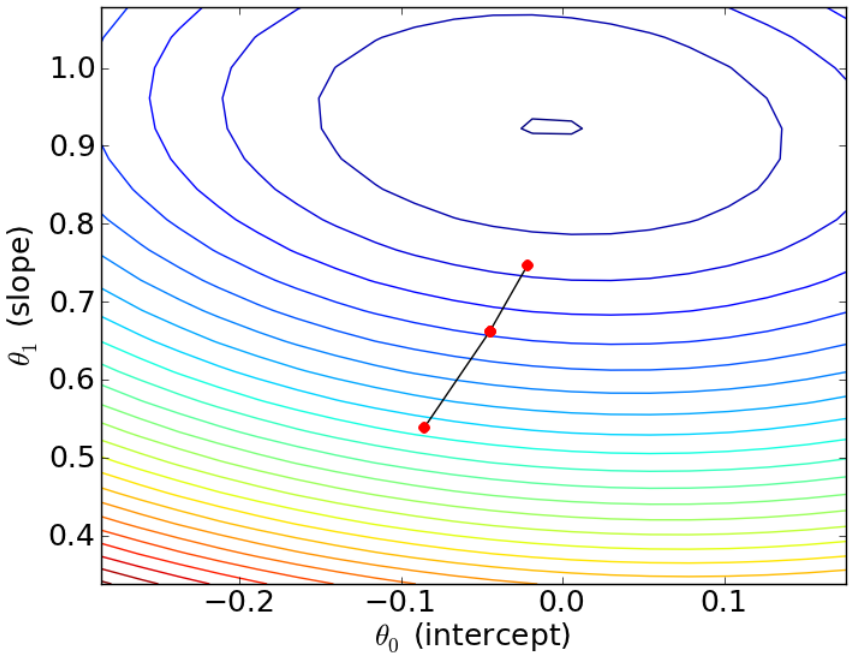

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
scaled y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
scaled x

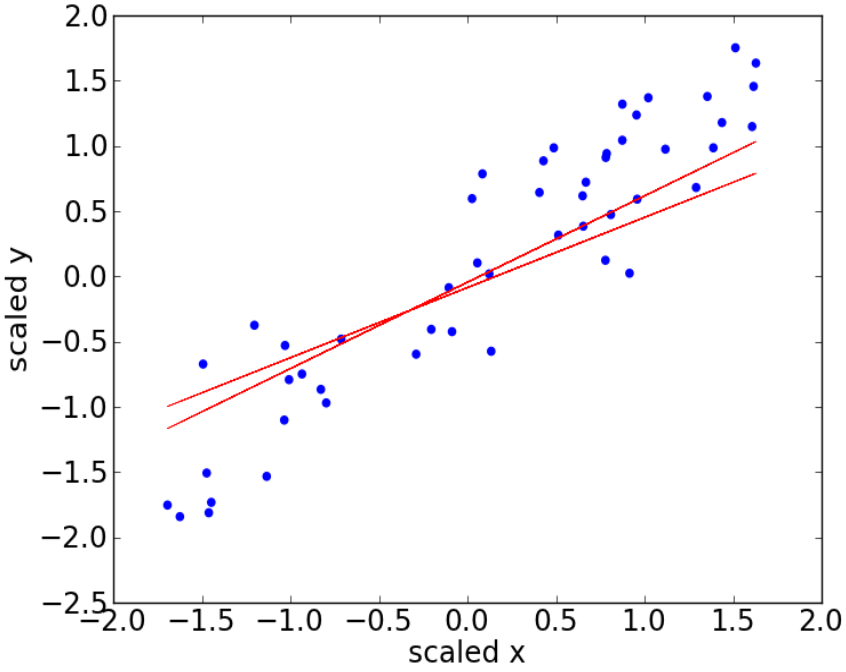


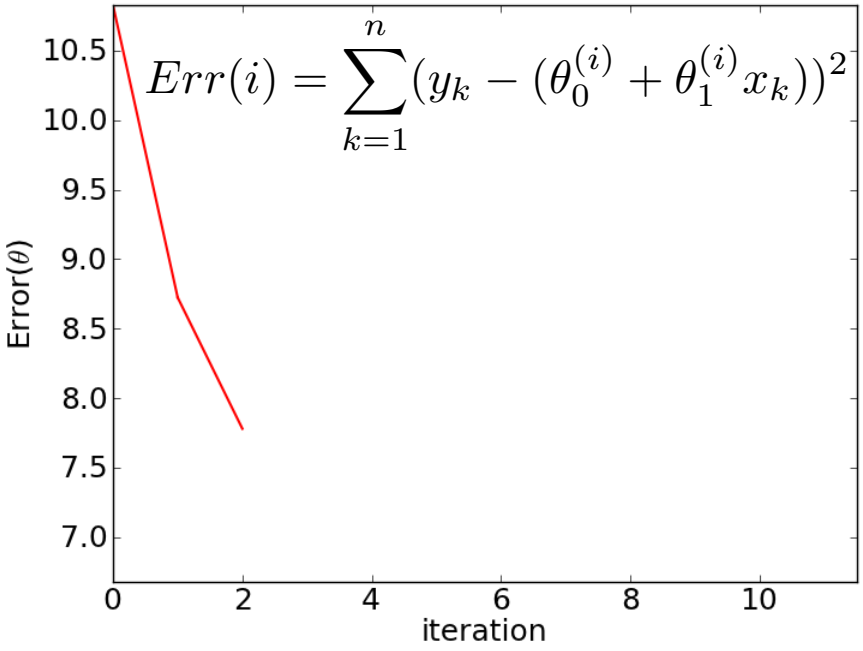




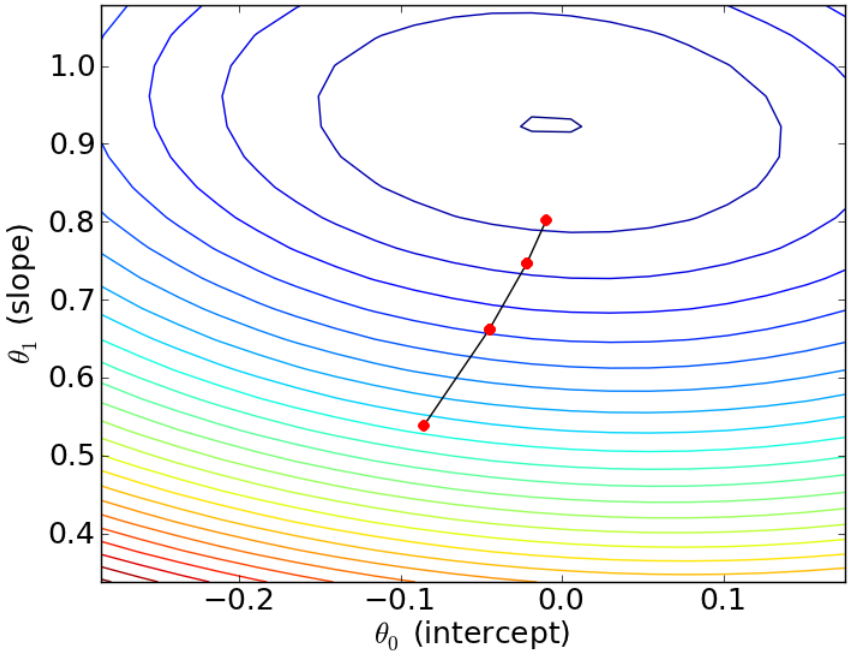
$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

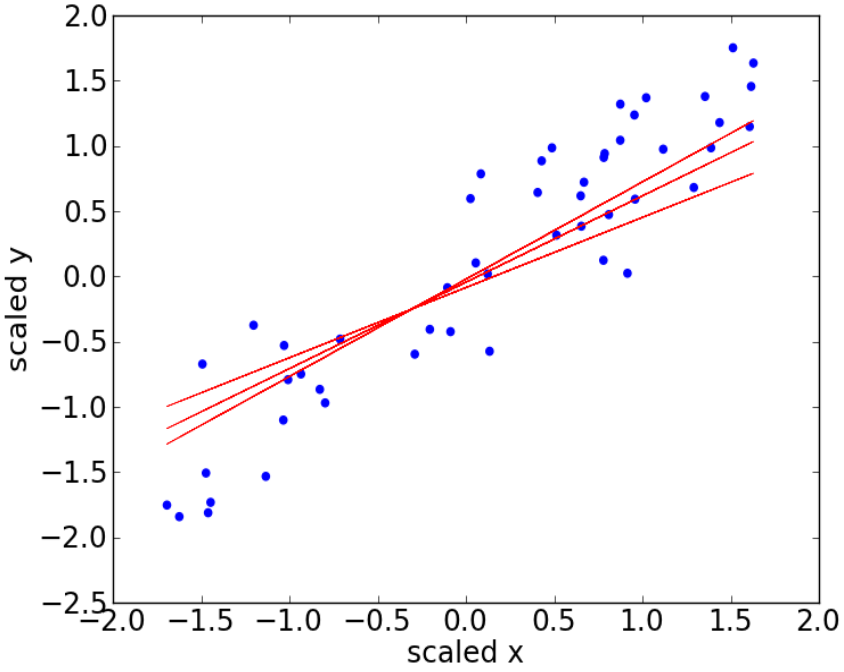


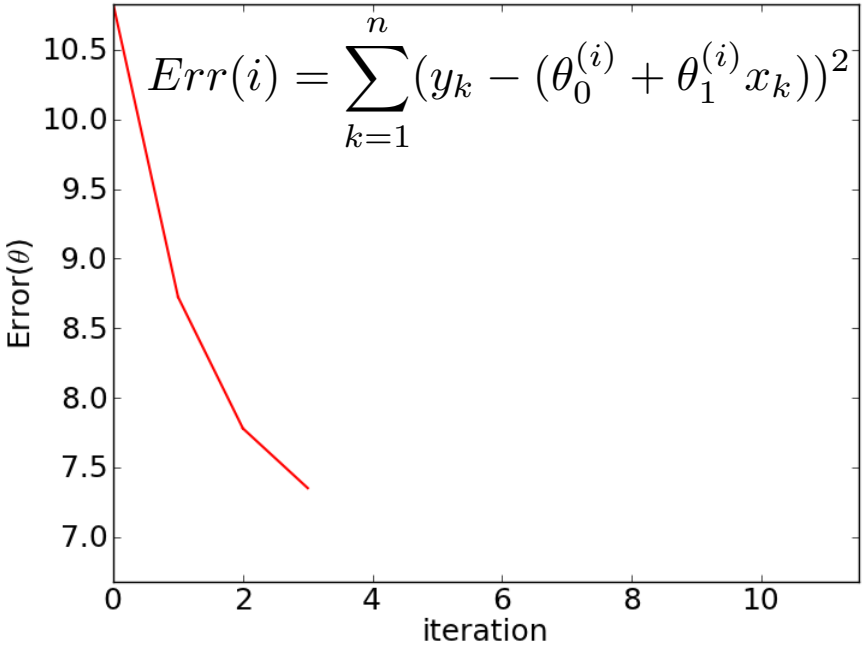




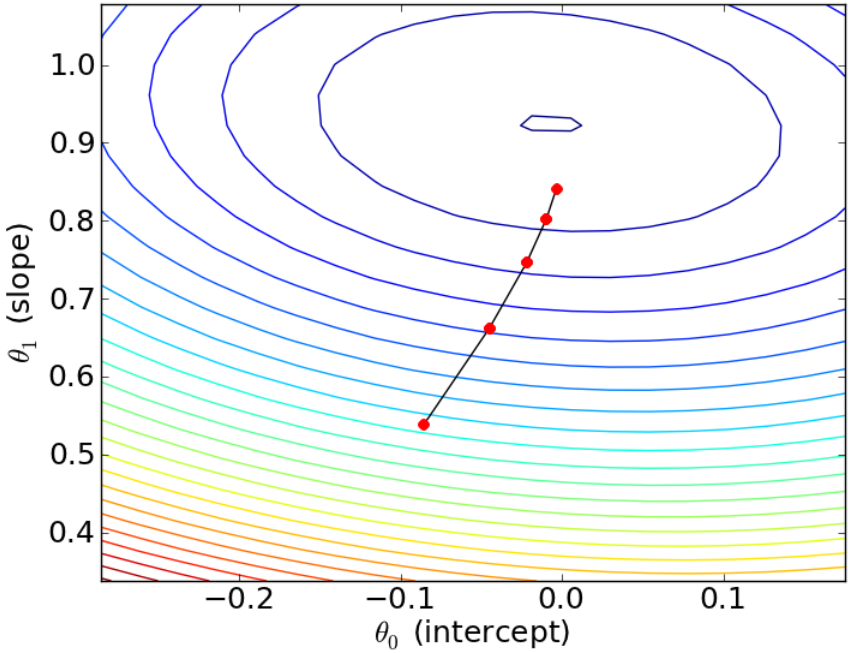
$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

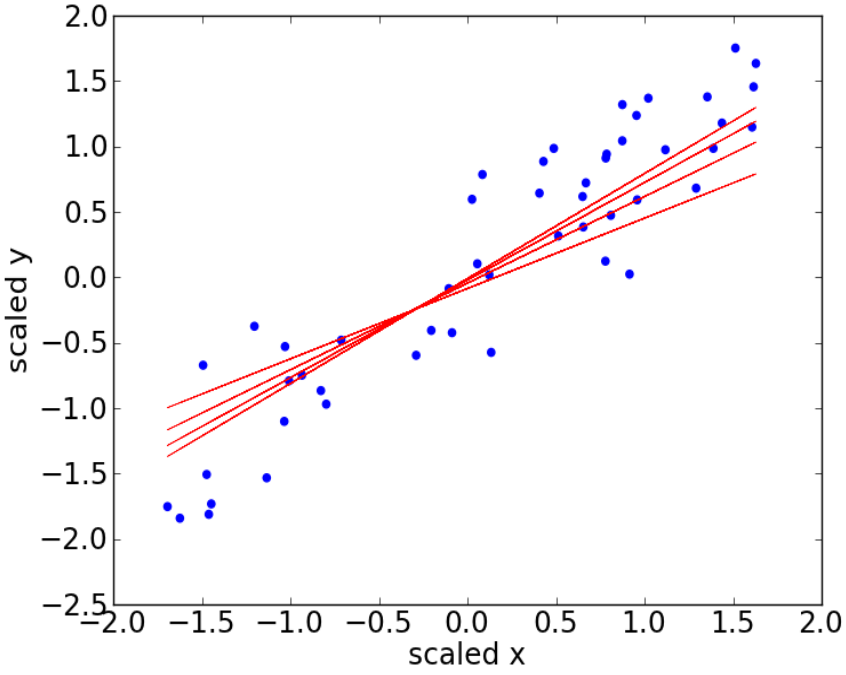


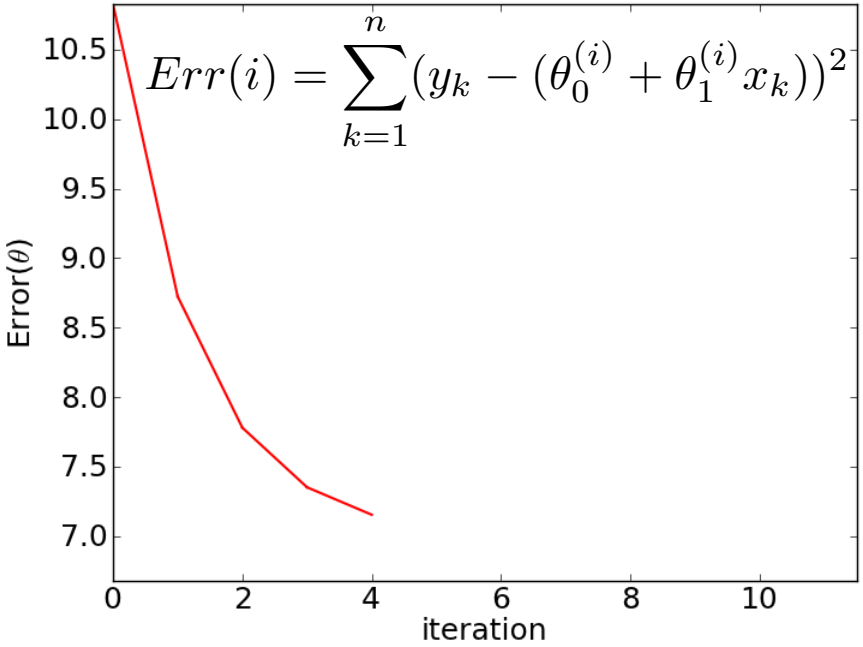




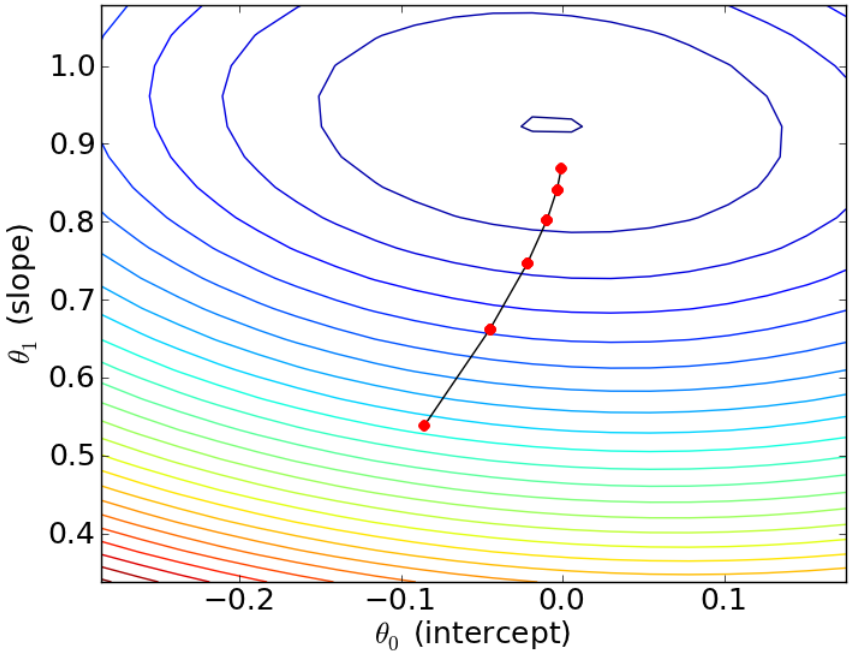
$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

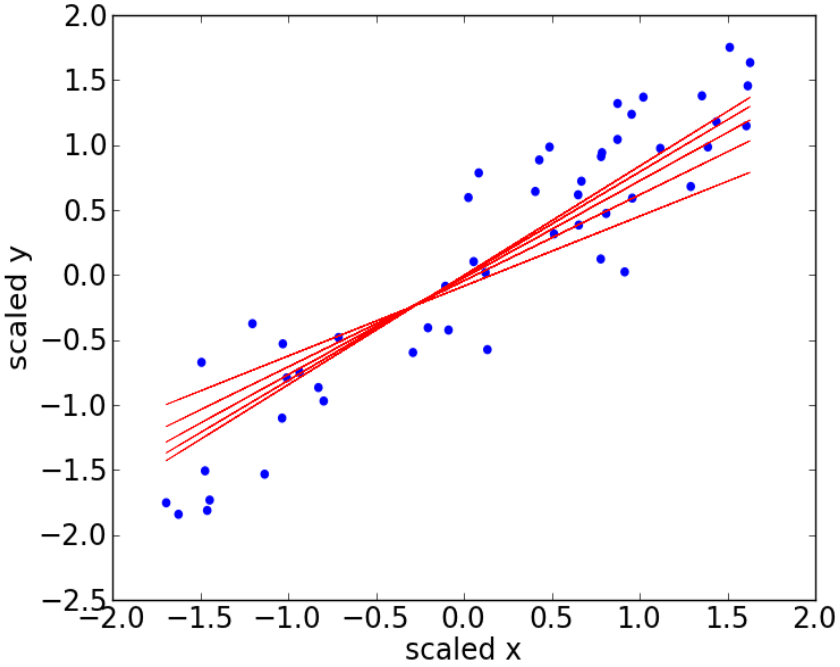


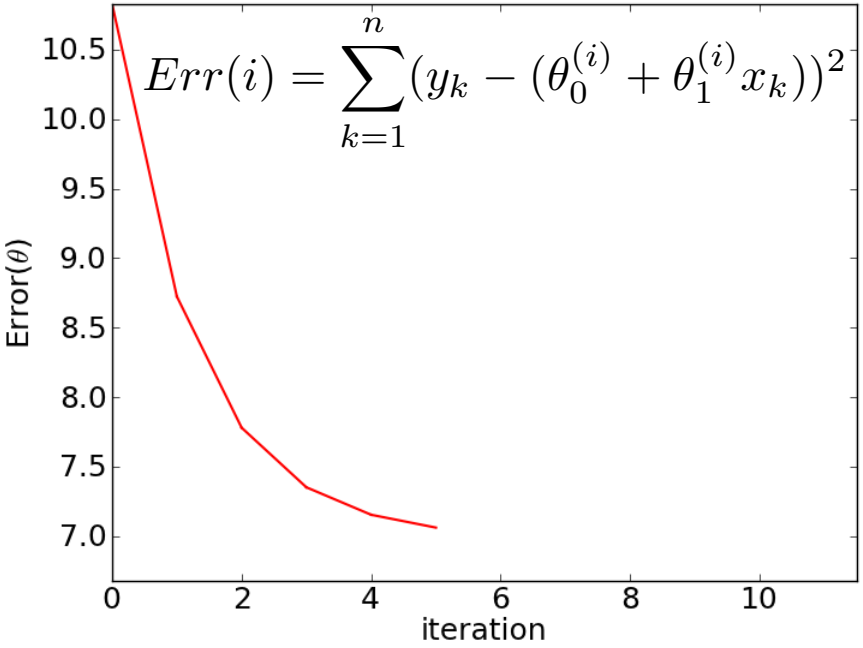




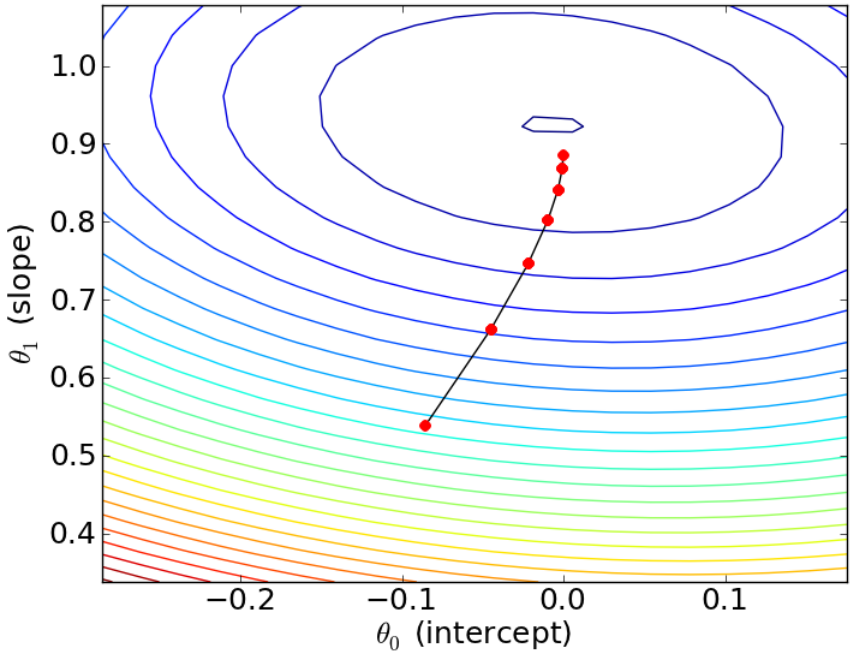
$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

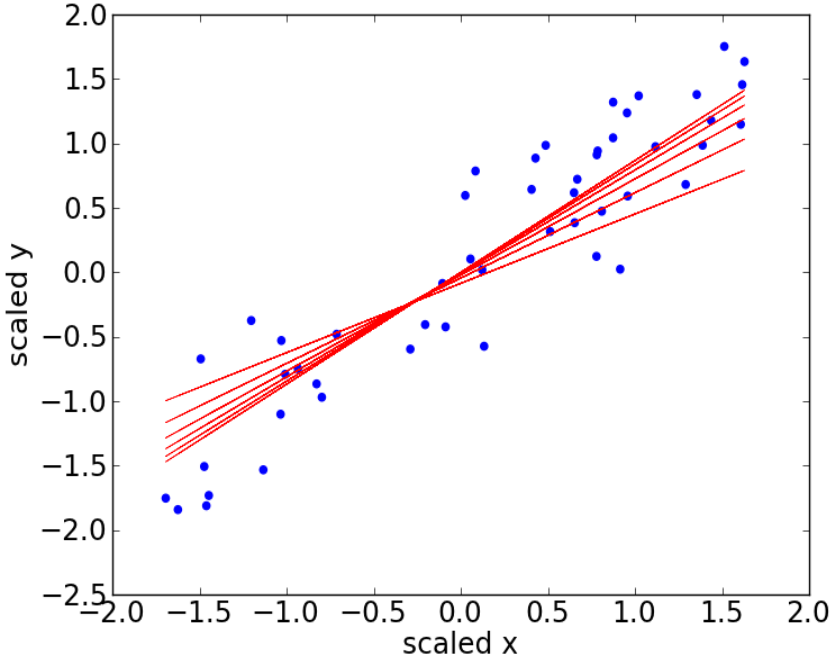


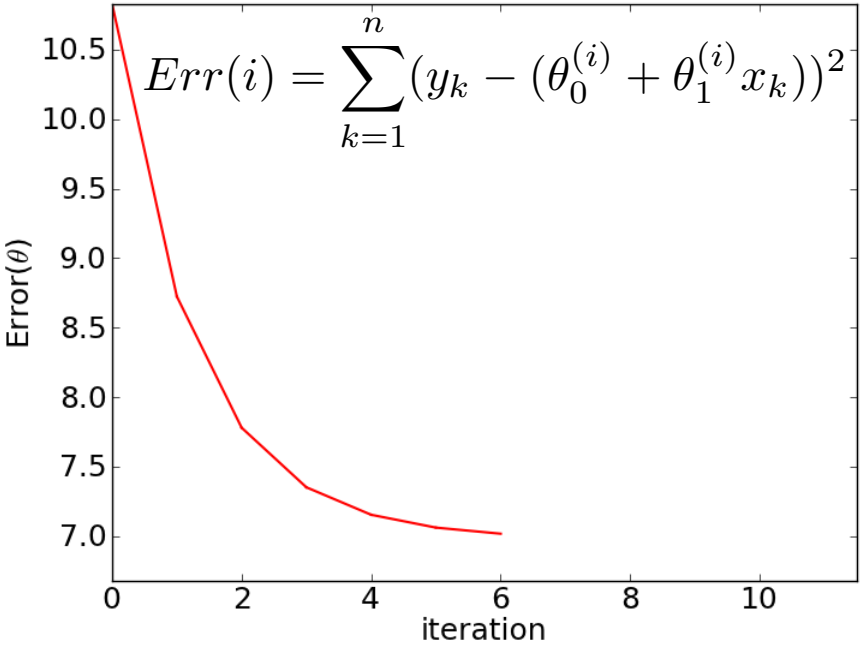




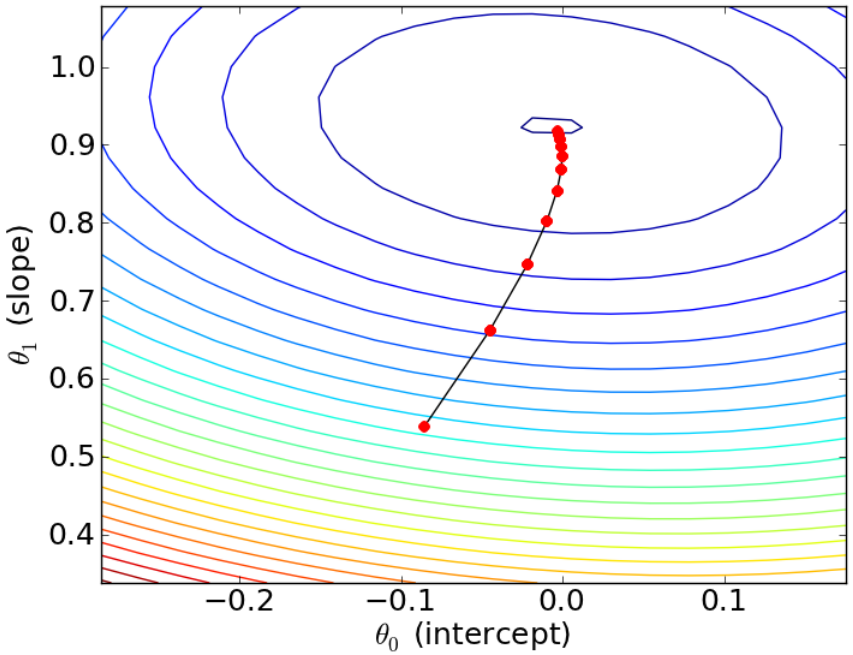
$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

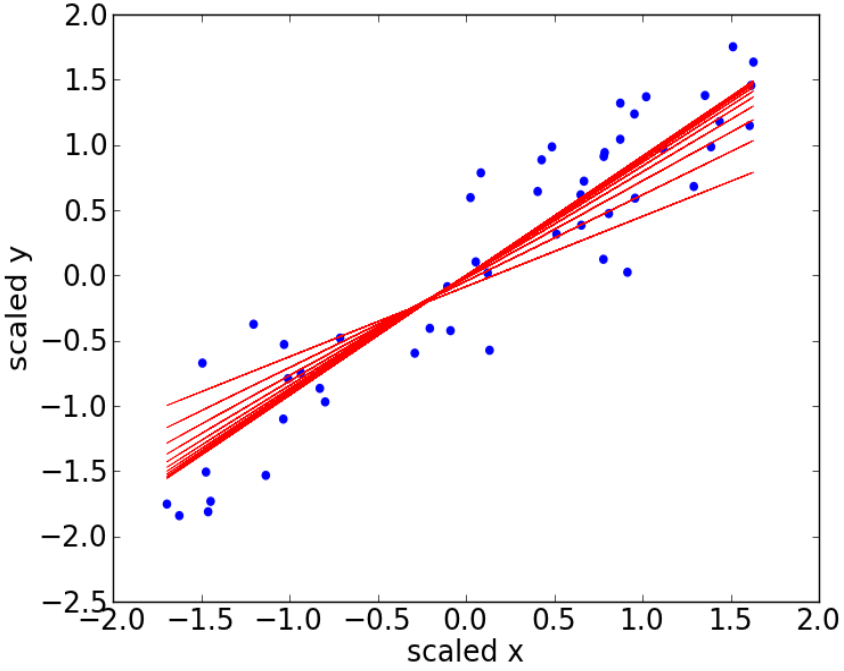


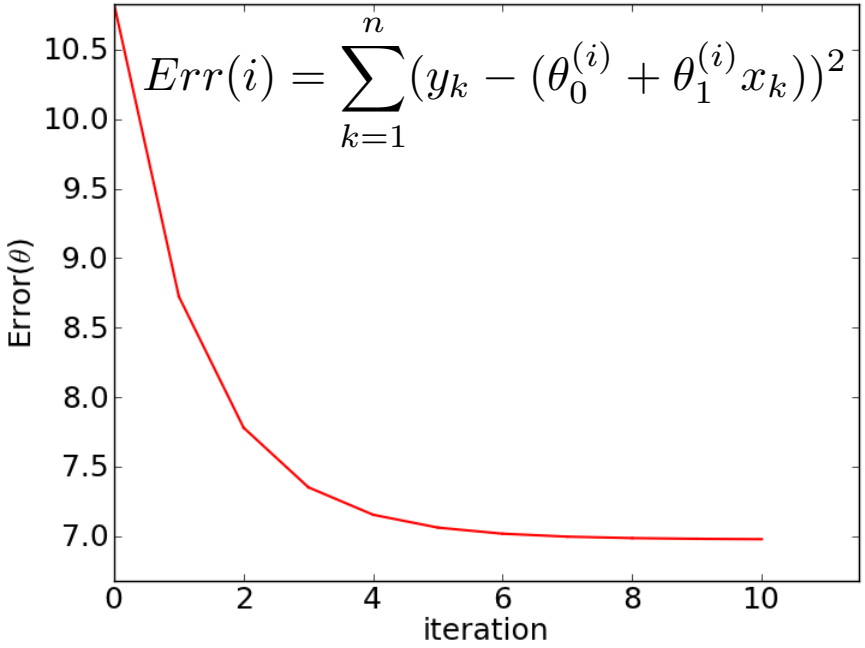




$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$



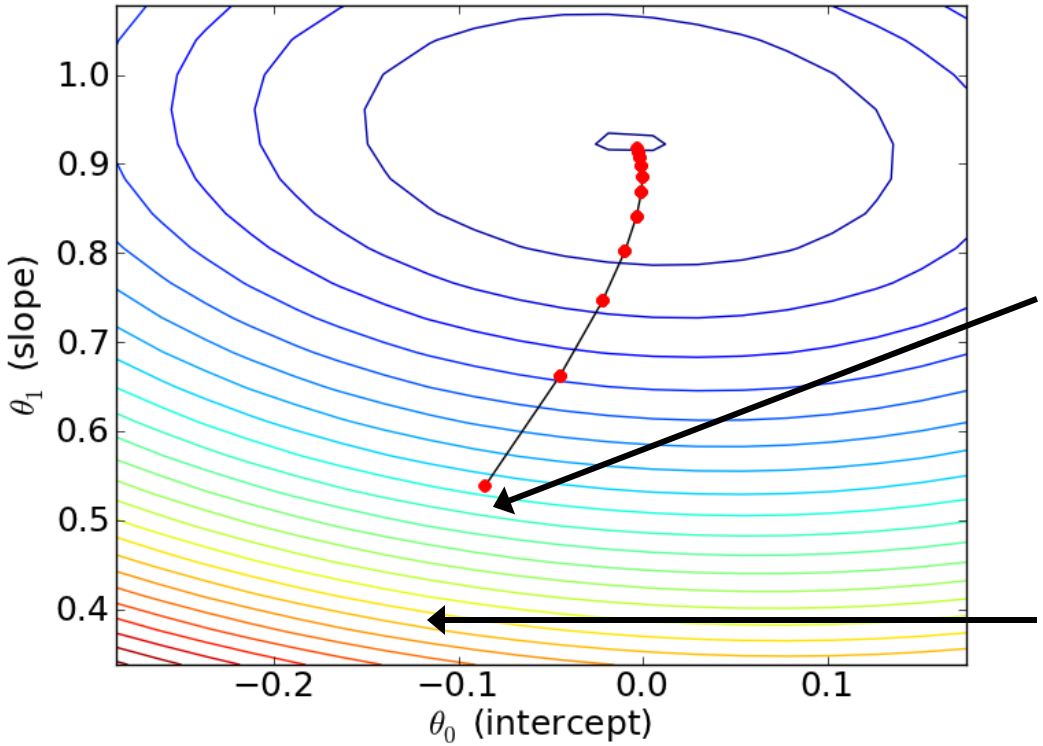




$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

$$\theta_0^{(i+1)} \leftarrow \theta_0^{(i)} + \alpha \frac{\delta}{\delta\theta_0} J(\theta^{(i)})$$

$$\theta_1^{(i+1)} \leftarrow \theta_1^{(i)} + \alpha \frac{\delta}{\delta\theta_1} J(\theta^{(i)})$$

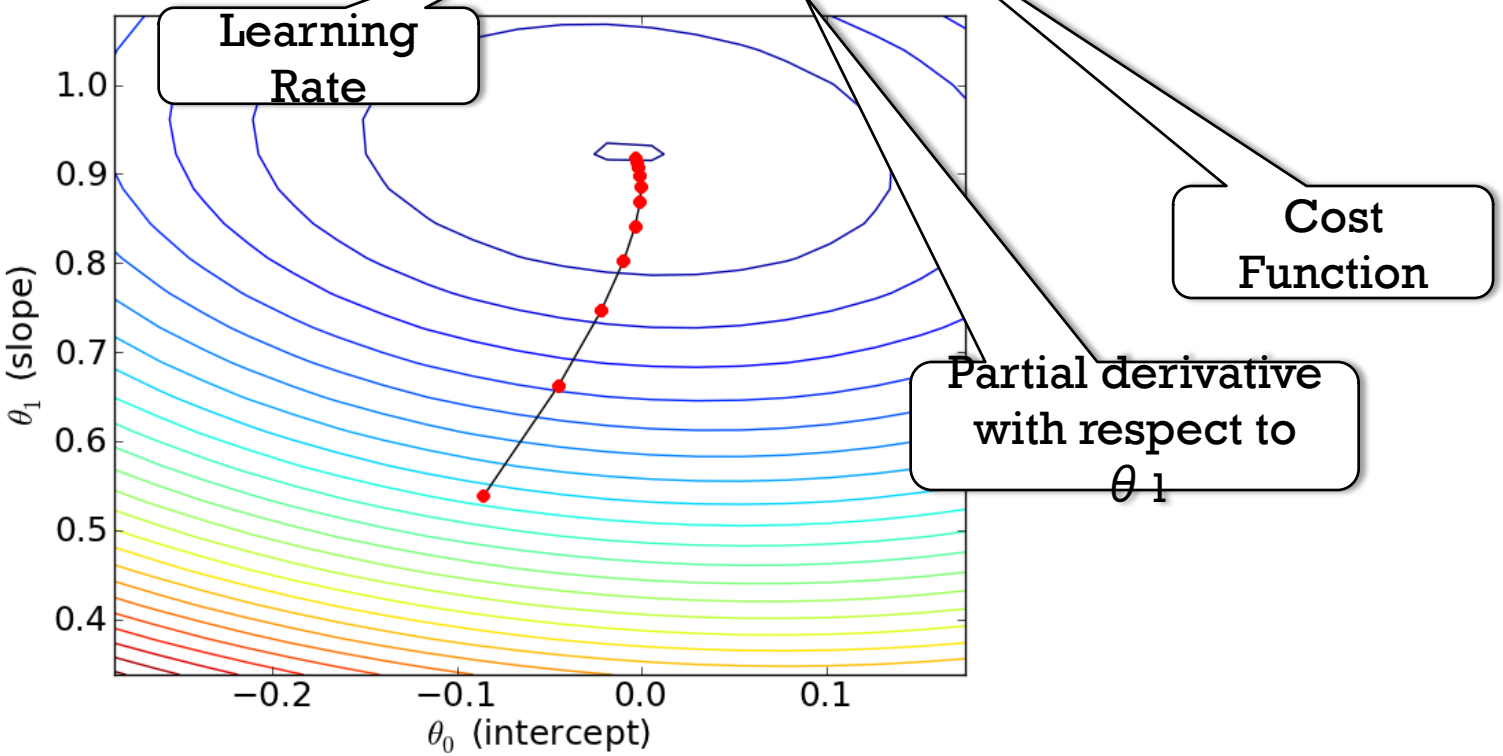


$(\theta_0^{(2)}, \theta_1^{(2)})$

$(\theta_0^{(1)}, \theta_1^{(1)})$

$$\theta_0^{(i+1)} \leftarrow \theta_0^{(i)} + \alpha \frac{\delta}{\delta\theta_0} J(\theta^{(i)})$$

$$\theta_1^{(i+1)} \leftarrow \theta_1^{(i)} + \alpha \frac{\delta}{\delta\theta_1} J(\theta^{(i)})$$



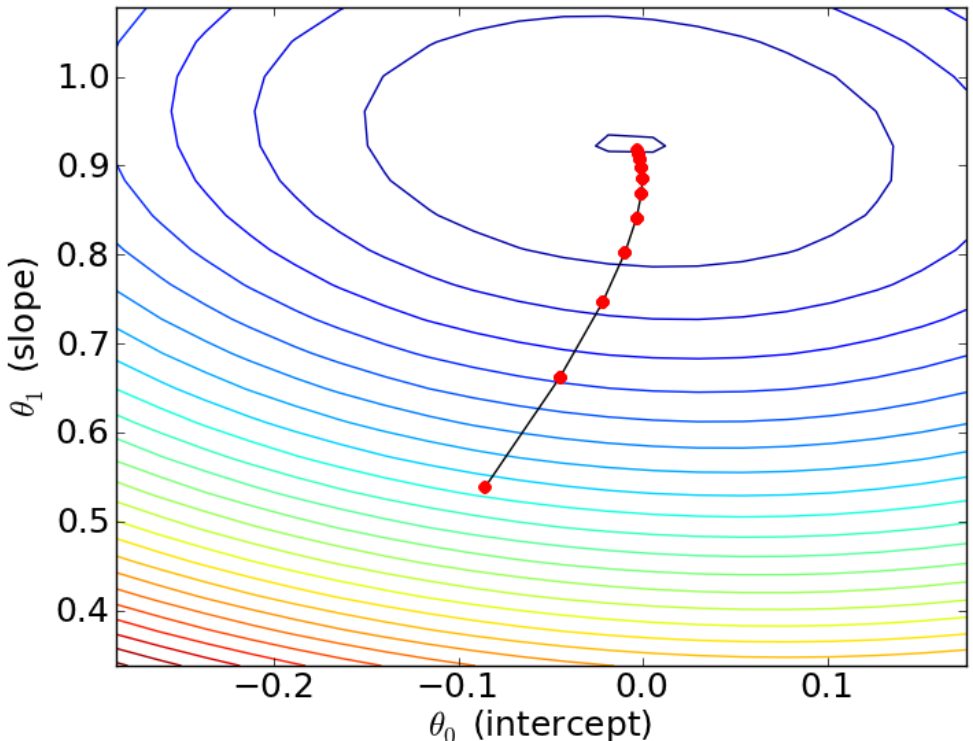
$$\theta_0^{(i+1)} \leftarrow \theta_0^{(i)} + \alpha \frac{\delta}{\delta\theta_0} \frac{1}{2} \sum_{k=1}^{n} (h^{(i)}(x_k) - y_k)^2$$

Model at *i*

Learning Rate

Gradient

Cost Function



$$\frac{\delta}{\delta\theta_1}\frac{1}{2}\sum_{k=1}^{n}(h(x_k)-y_k)^2$$

$$\sum_{k=1}^{n}2\frac{1}{2}(h(x_k)-y_k)\frac{\delta}{\delta\theta_1}(h(x_k)-y_k)$$

$$\sum_{k=1}^{n}(h(x_k)-y_k)\frac{\delta}{\delta\theta_1}(\theta_0+\theta_1 x_k-y_k)$$

$$\sum_{k=1}^{n}(h(x_k)-y_k)x_k$$

$$\theta_1^{(i+1)} \leftarrow \theta_1^{(i)} + \alpha \frac{\delta}{\delta \theta_0} \frac{1}{2} \sum_{k=1}^{n} (h^{(i)}(x_k) - y_k)^2$$

$$\theta_1^{(i+1)} \leftarrow \theta_1^{(i)} + \alpha \sum_{k=1}^{n} (\theta_0^{(i)} + \theta_1^{(i)}(x_k) - y_k) x_k$$

while not converged:

$$\theta_0^{(i+1)} \leftarrow \theta_0^{(i)} + \alpha \sum_{k=1}^{n} (\theta_0^{(i)} + \theta_1^{(i)}(x_k) - y_k) * 1.0$$

$$\theta_1^{(i+1)} \leftarrow \theta_1^{(i)} + \alpha \sum_{k=1}^{n} (\theta_0^{(i)} + \theta_1^{(i)}(x_k) - y_k) x_k$$

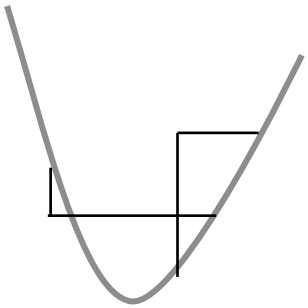
SOME QUESTIONS

Initialization

- Where do you drop the ball?
- "small random values"

Step size

- We don't really "roll," we "jump" in the direction of steepest descent
- How far should we jump?
- Too far and you might hop over the minimum and raise the function vaue
- Too small and performance is bad



WHERE WE ARE

Supervised Learning

Rules/Trees

Overfitting

Ensembles

Ex: Random Forests

k Nearest Neighbors

Next

- *Optimization with Gradient Descent*
- *Quick Intuition for Logistic Regression*
- *Quick Intuition for Support Vector Machines*
- *Regularization*
- *Stochastic Gradient Descent*

WHAT'S THE POINT?

$$\theta_j^{(i+1)} \leftarrow \theta_j^{(i)} + \alpha \frac{\delta}{\delta \theta_j} J(\theta^{(i)})$$

Model parameters can be anything

Cost function can be anything*

*needs to be differentiable

OTHER COST FUNCTIONS

vector of weights

vector of instance data

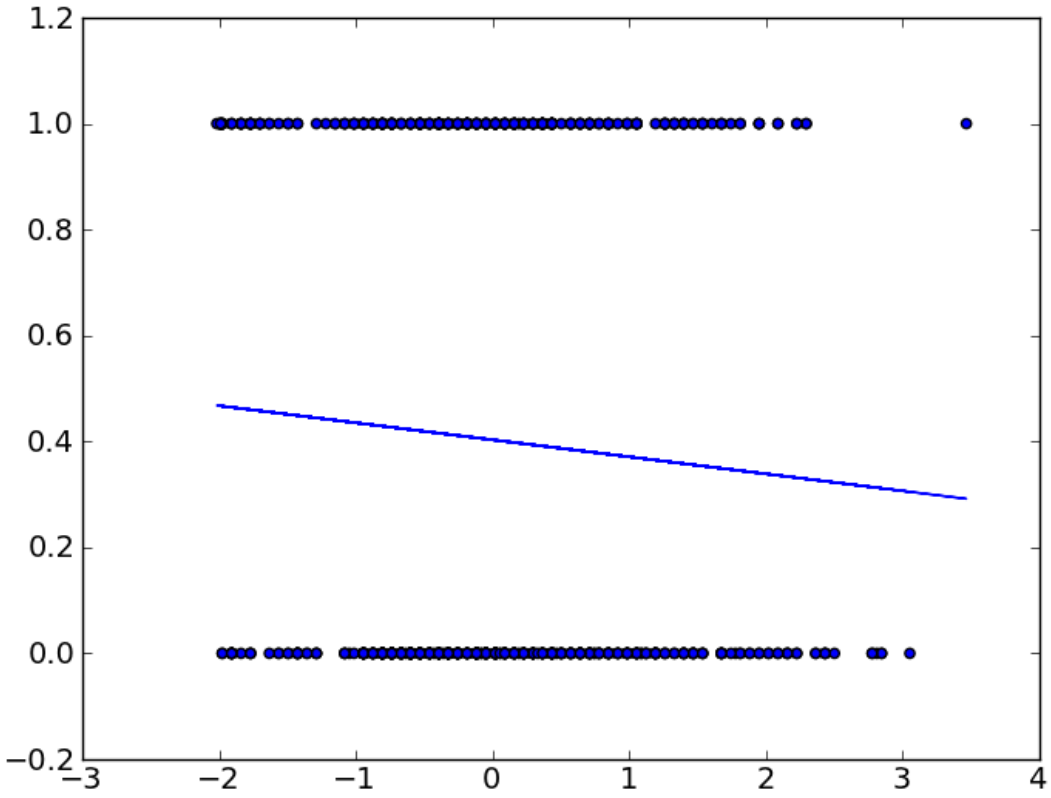
Logistic Regression

$$J(\theta) = \frac{1}{n}\sum_{i=0}^{n} \log_2\left(1 + \exp(-y_i(\theta \cdot x_i))\right) + \frac{\lambda}{2}||\theta||^2$$

Support Vector Machines

$$J(\theta) = \frac{1}{n}\sum_{i=0}^{n} \max(1 - y_i(\theta \cdot x_i), 0) + \frac{\lambda}{2}||\theta||^2$$

QUICK INTUITION FOR LOGISTIC REGRESSION (1)



Gradient descent regression line for titanic data.

Predicting survival (y-axis) from (normalized) age (x-axis).

What does this line mean?

Nothing really

QUICK INTUITION FOR LOGISTIC REGRESSION (2)

$$f(x) = \frac{e^x}{e^x + 1}$$

Maps **any** number to the range (0,1)

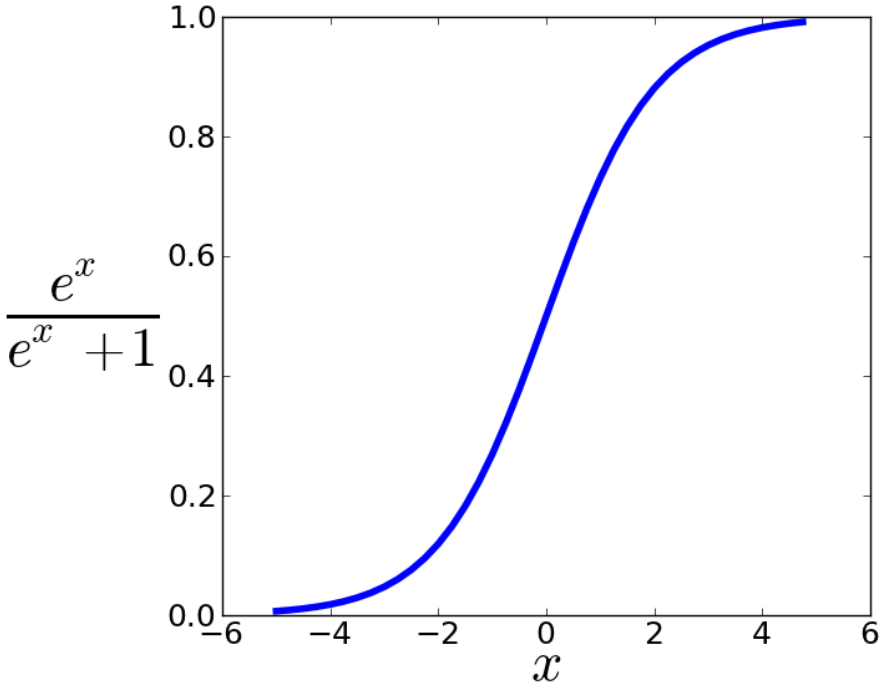
Interpret the result as a probability

Interpret categorical classes numerically

What is the *probability* a passenger survived?

The cost function is constructed to maximize the probability of correct classification.

Easy to work with



WHERE WE ARE

Supervised Learning

Rules/Trees

Overfitting

Ensembles

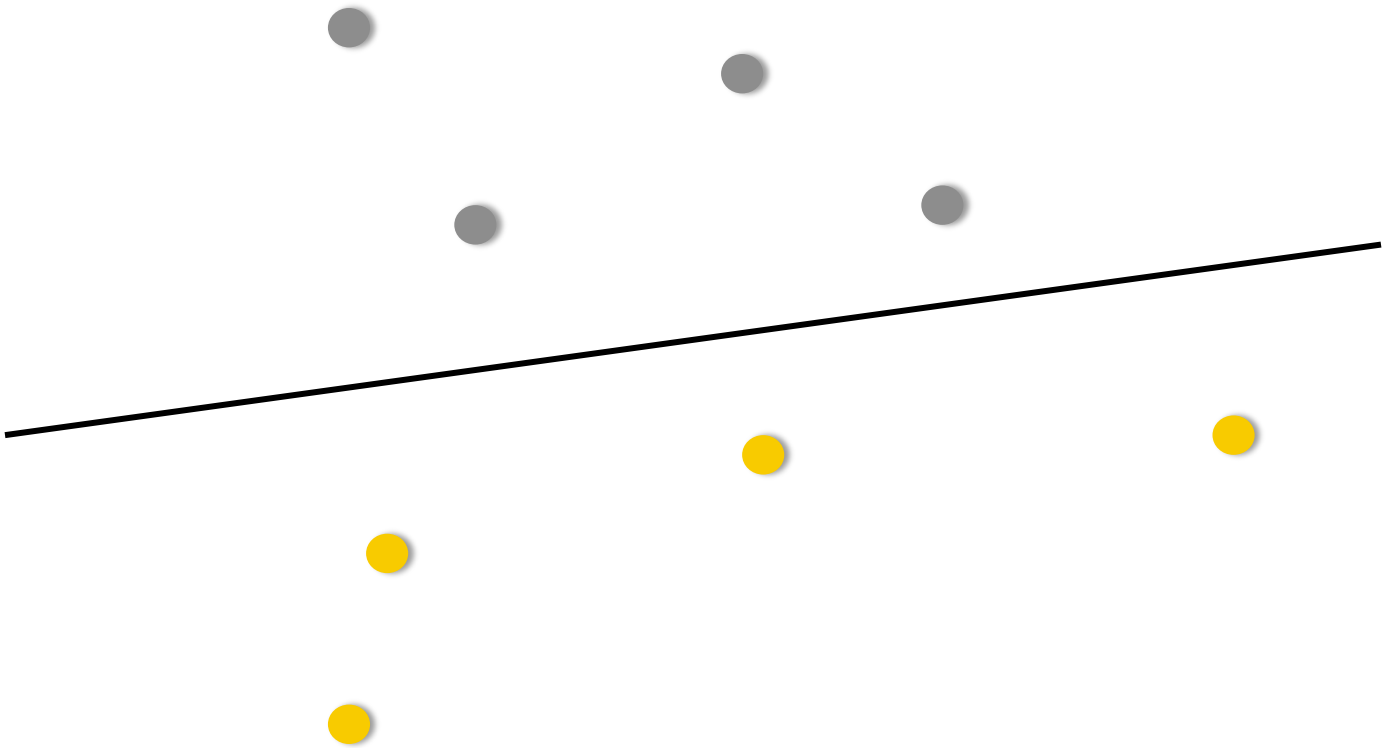
Ex: Random Forests

k Nearest Neighbors

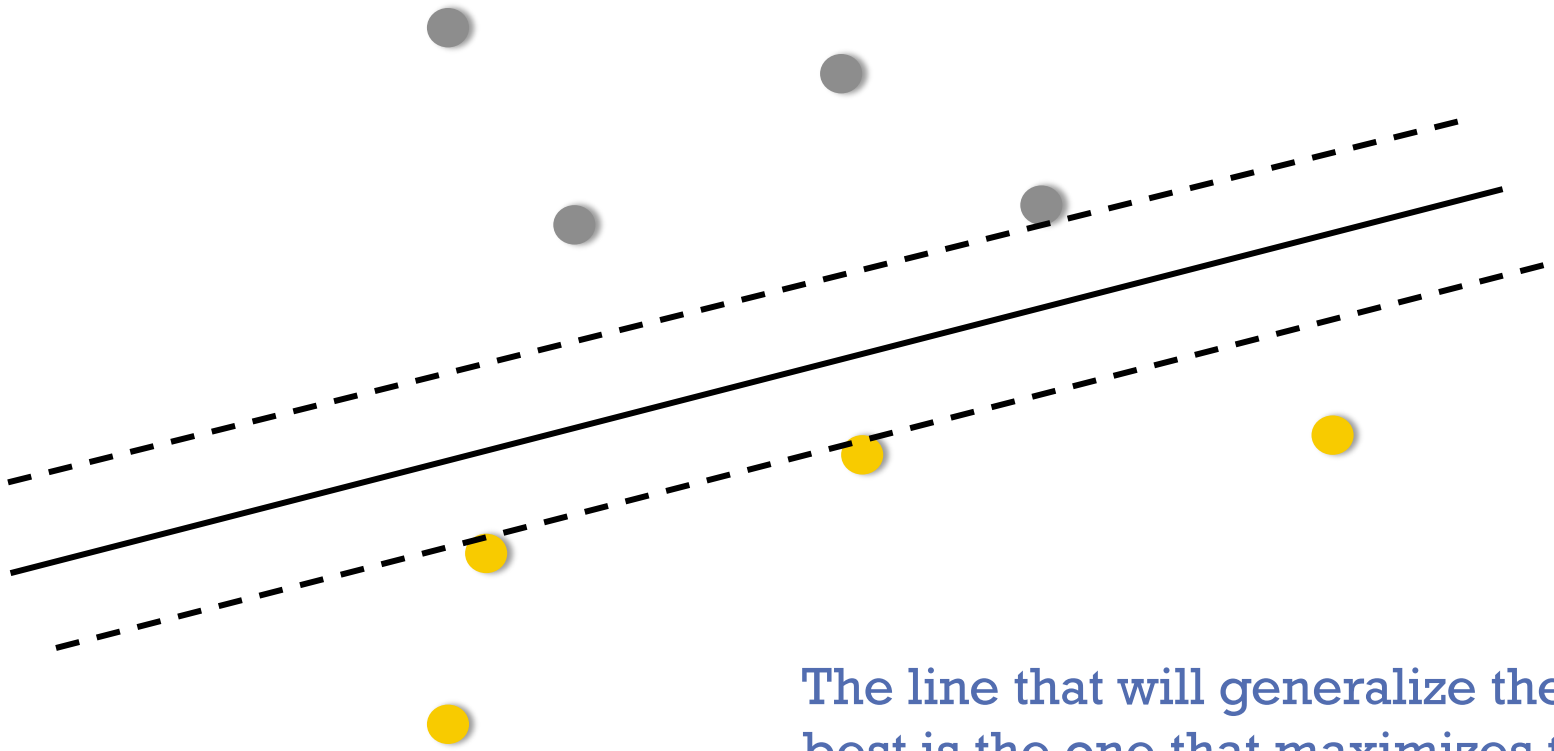
Next

- *Optimization with Gradient Descent*
- *Quick Intuition for Logistic Regression*
- *Quick Intuition for Support Vector Machines*
- *Regularization*
- *Stochastic Gradient Descent*

QUICK INTUITION FOR SUPPORT VECTOR MACHINES (1)

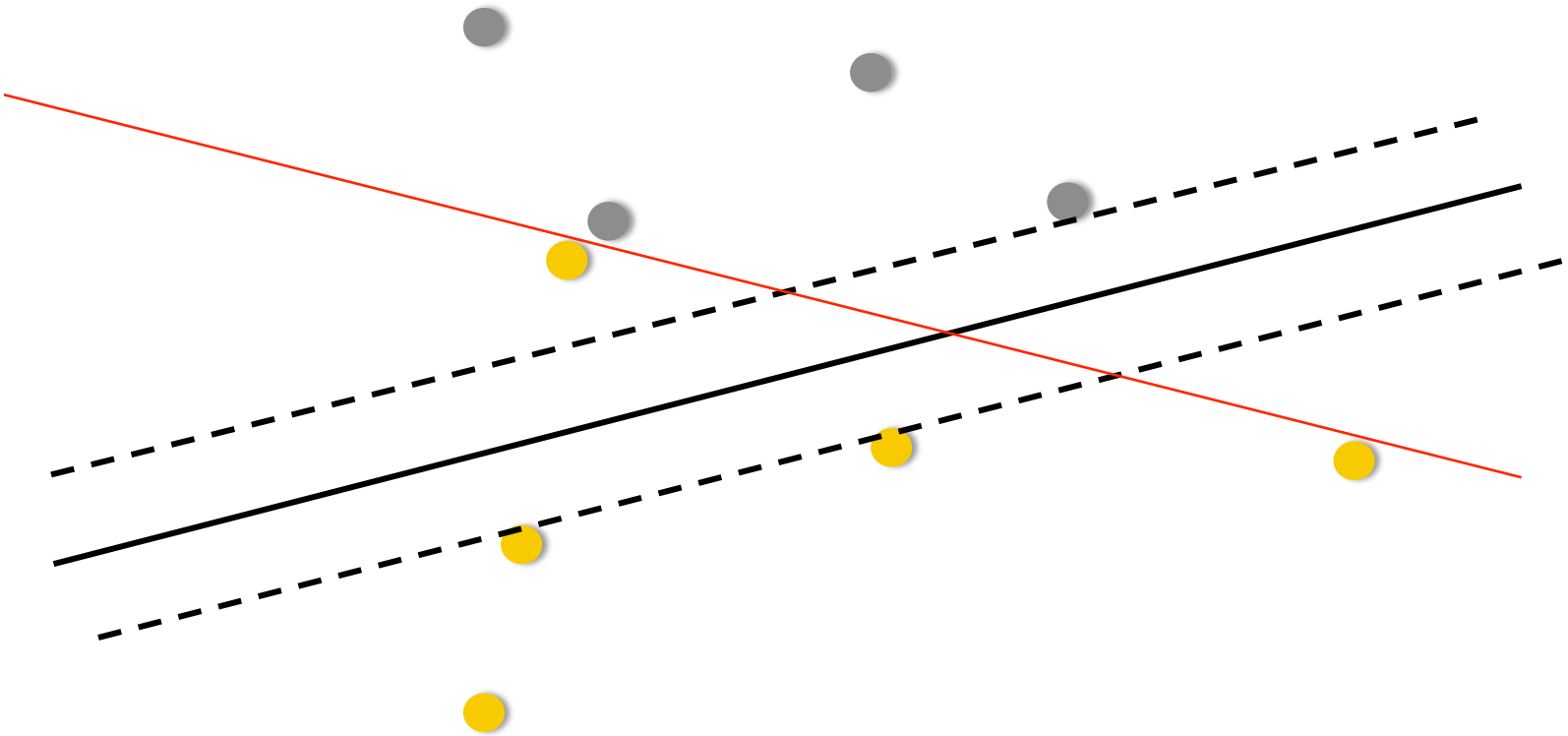


QUICK INTUITION FOR SUPPORT VECTOR MACHINES (2)



The line that will generalize the best is the one that maximizes the margin between the classes.

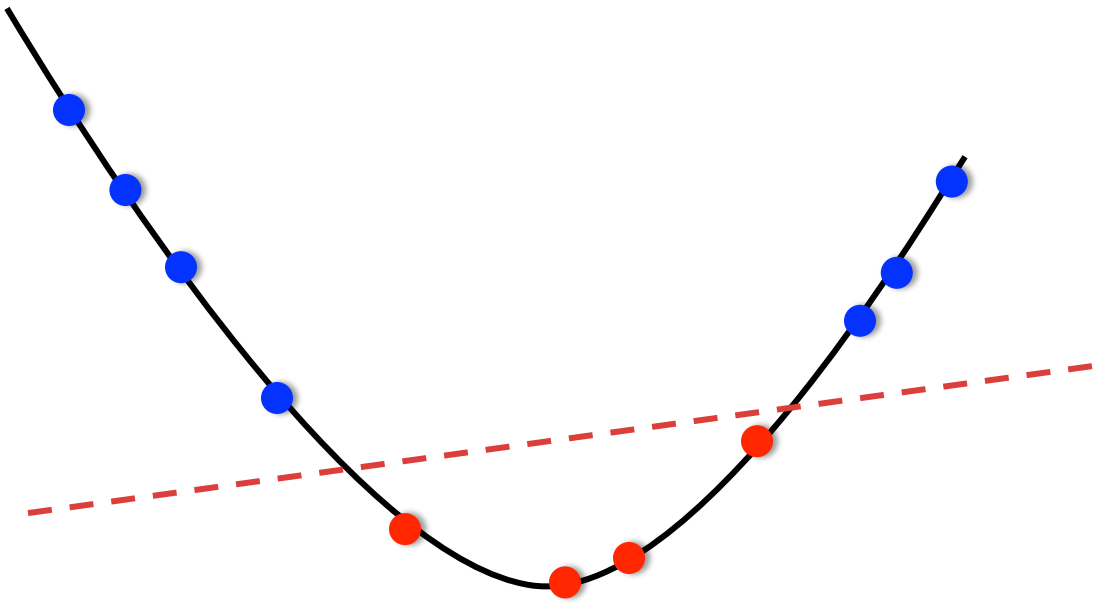
QUICK INTUITION FOR SUPPORT VECTOR MACHINES (3)



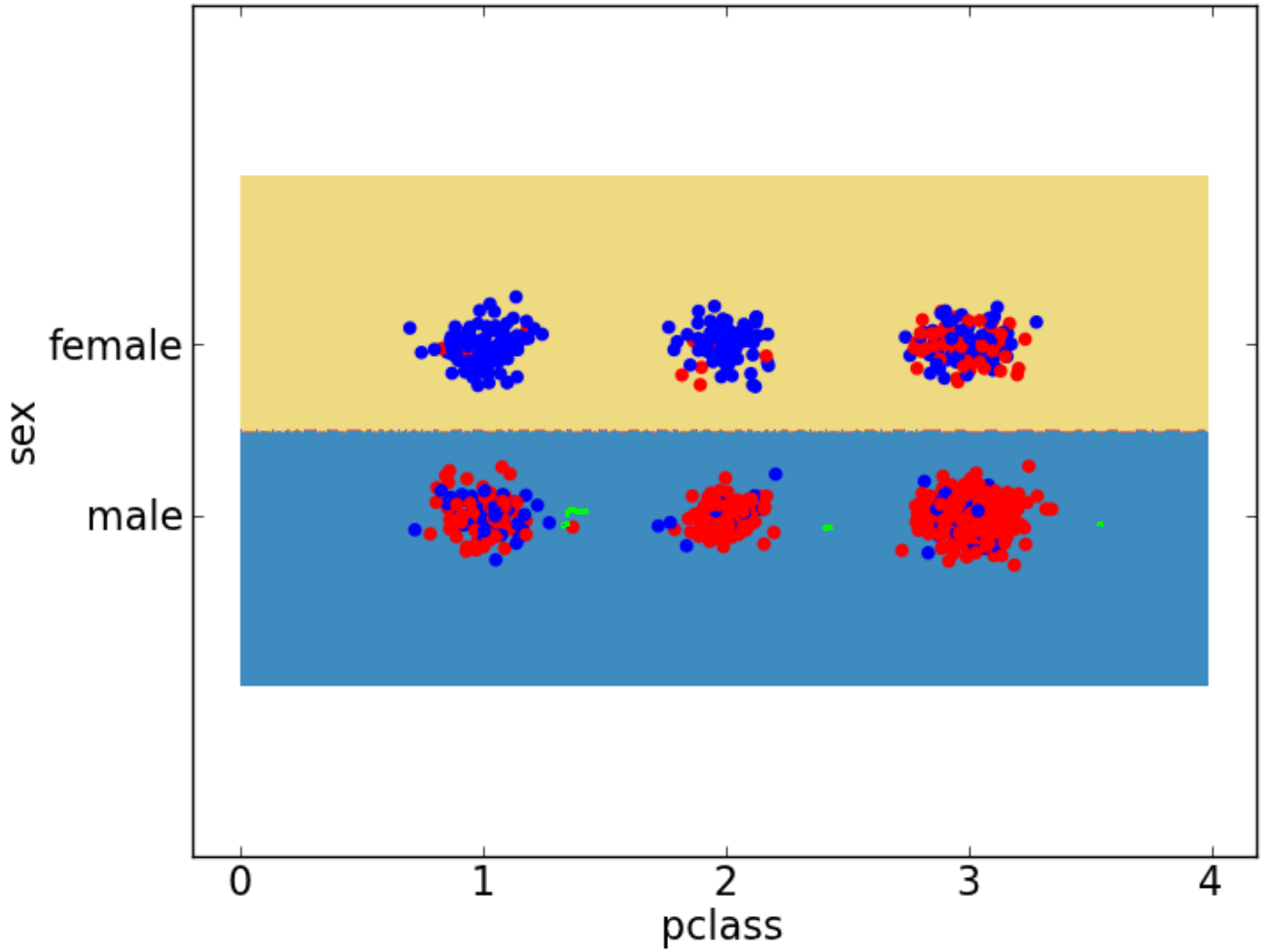
QUICK INTUITION FOR SUPPORT VECTOR MACHINES (4)

Not linearly separable in 1D:

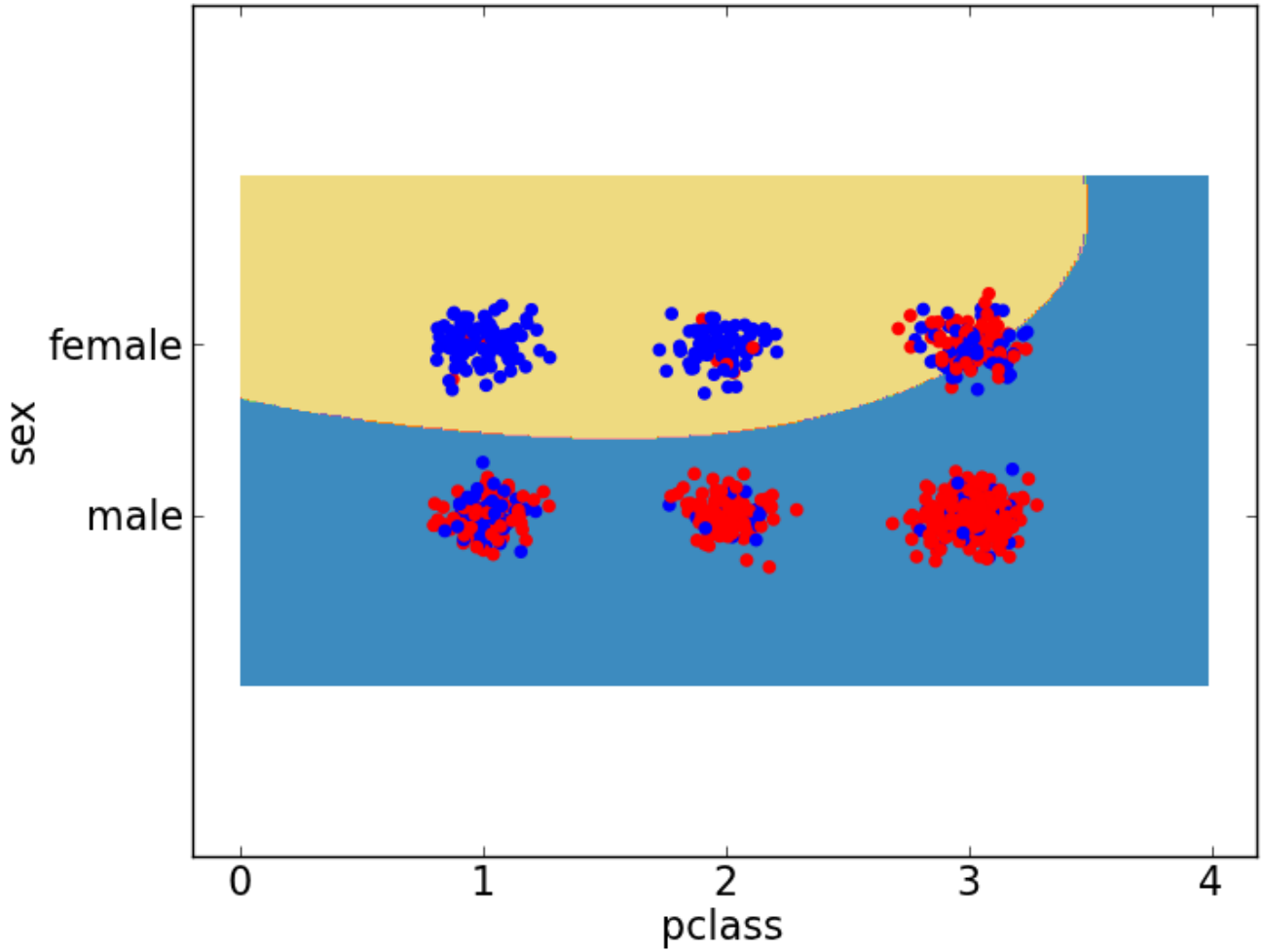
Map the data into a higher dimensional space by applying a kernel



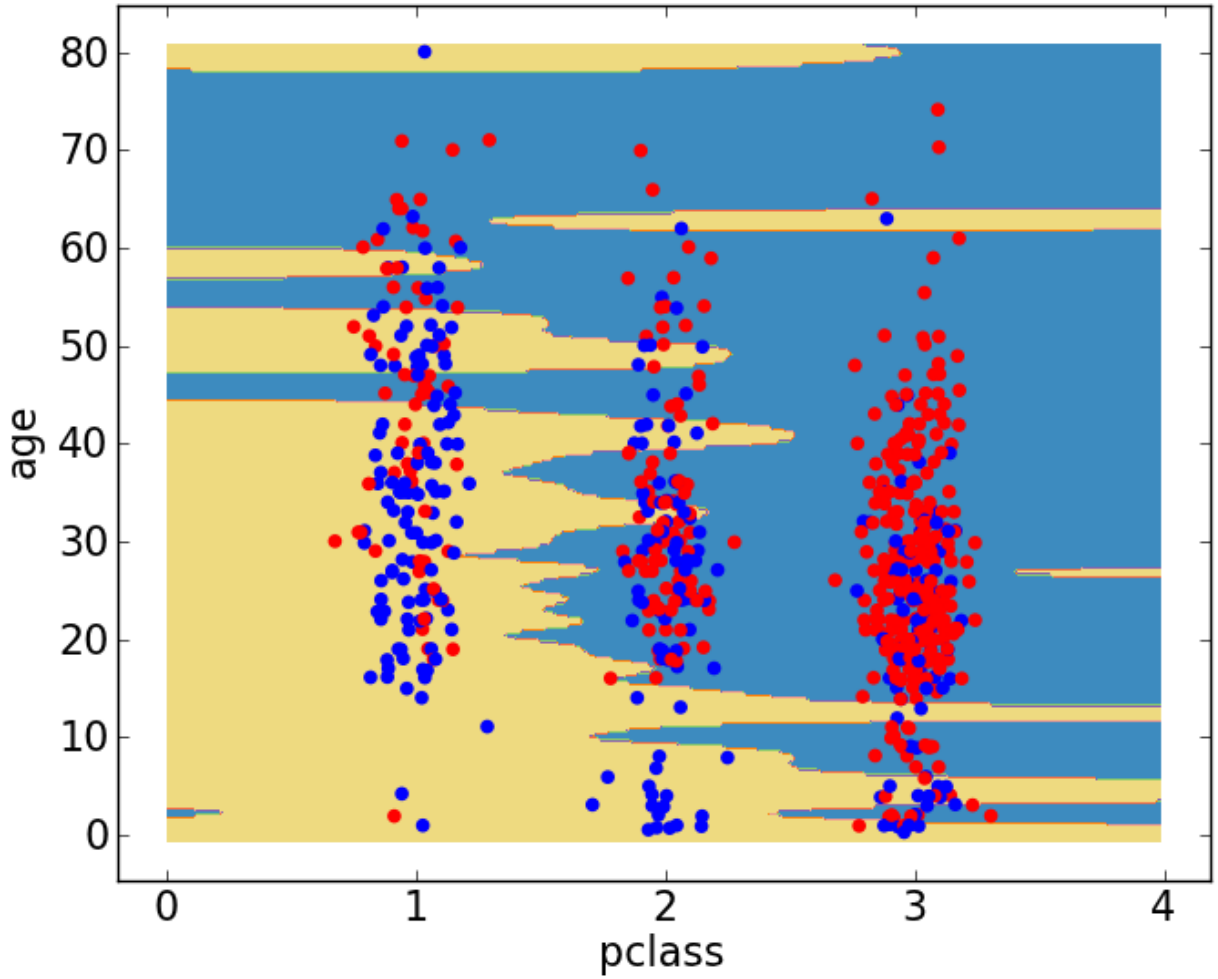
SUPPORT VECTOR MACHINE MODEL, TITANIC DATA, LINEAR KERNEL



SUPPORT VECTOR MACHINE MODEL, TITANIC DATA, RADIAL BASIS FUNCTION KERNEL

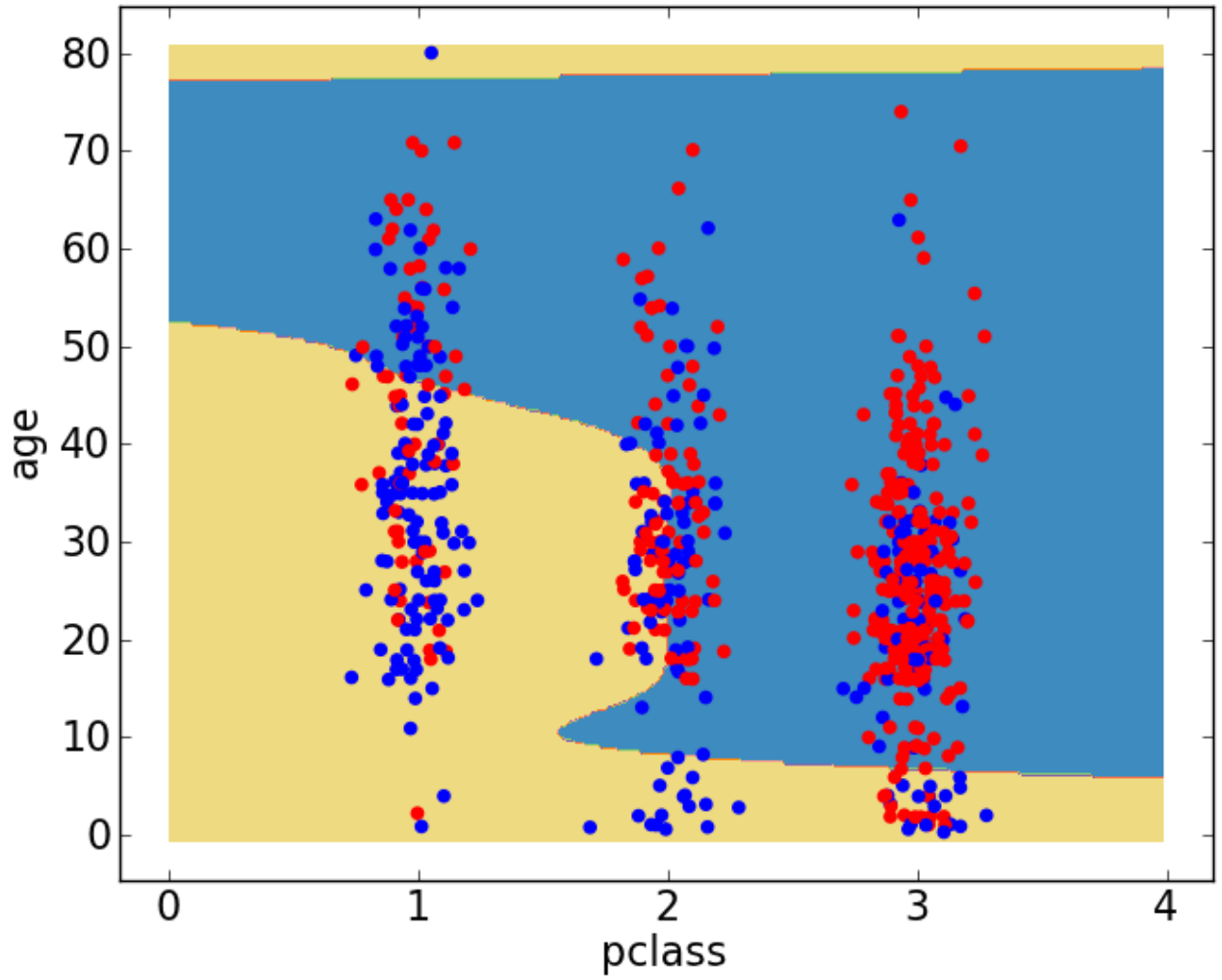


SUPPORT VECTOR MACHINE MODEL, TITANIC DATA, RADIAL BASIS FUNCTION KERNEL



overfitting?

SUPPORT VECTOR MACHINE MODEL, TITANIC DATA, RADIAL BASIS FUNCTION KERNEL



a lower gamma, a parameter that controls that balances model complexity against accuracy

WHERE WE ARE

Supervised Learning

Rules/Trees

Overfitting

Ensembles

Ex: Random Forests

k Nearest Neighbors

Next

- *Optimization with Gradient Descent*
- *Quick Intuition for Logistic Regression*
- *Quick Intuition for Support Vector Machines*
- *Regularization*
- *Stochastic Gradient Descent*

BACK TO THE COST FUNCTIONS

Regularization term

Logistic Regression

$$J(\theta) = \frac{1}{n}\sum_{i=0}^{n} \log_2\left(1 + \exp(-y_i(\theta \cdot x_i))\right) + \frac{\lambda}{2}||\theta||^2$$

Support Vector Machines

$$J(\theta) = \frac{1}{n}\sum_{i=0}^{n} \max(1 - y_i(\theta \cdot x_i), 0) + \frac{\lambda}{2}||\theta||^2$$

QUICK INTUITION FOR REGULARIZATION

Consider high-dimensional problems

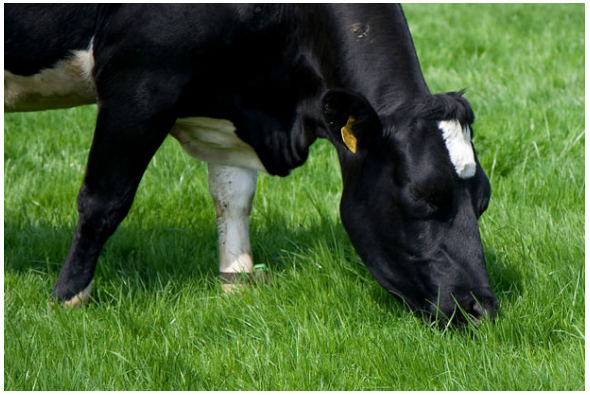
- Image recognition

  one weight per pixel

- Document vectors

  500k term weights

With so many weights, likely that many are correlated



Nearby pixels are highly correlated.

As one weight goes up, another goes down to compensate.

And so weights may explode – overfitting again

Need to enforce some condition on the weights to prefer simple models.

The regularization term provides this balance

ASIDE ON NORMS

Norm: any function that assigns a strictly positive number to every non-zero vector

$$L^p\text{-norm} = ||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

ASIDE ON COST FUNCTIONS

not a norm $\sum_i H_i - H'_i$

errors cancel out;
usually not what you want

L^1-norm $\sum_i |H_i - H'_i|$

Asserts that 1 error of 7 units is as bad as 7 errors of 1 unit each

L^2-norm $\sqrt{\sum_i (H_i - H'_i)^2}$

Asserts that 1 error of 7 units is as bad as 49 errors of 1 unit each

$\frac{1}{n} \sum_i^n (H_i - H'_i)^2$

Average squared error per data point; useful when comparing methods that filter the data differently

BACK TO REGULARIZATION

$$||\theta||_1 = \sum_i |\theta_i|$$

"LASSO"
Regularized Least Squares; L1 norm

$$\alpha \sum_{k=1}^{n} (h(x_k) - y_k)^2 + \frac{\lambda}{2} ||\theta||_1$$

$$||\theta||_2 = \sqrt{\sum_i \theta_i^2}$$

"Ridge Regression"
Regularized Least Squares; L2 norm

$$\alpha \sum_{k=1}^{n} (h(x_k) - y_k)^2 + \frac{\lambda}{2} ||\theta||_2^2$$

WHERE WE ARE

Supervised Learning

Rules/Trees

Overfitting

Ensembles

Ex: Random Forests

k Nearest Neighbors

Next

- *Optimization with Gradient Descent*
- *Quick Intuition for Logistic Regression*
- *Quick Intuition for Support Vector Machines*
- *Regularization*
- *Stochastic Gradient Descent*

BACK TO GRADIENT DESCENT

We process the entire dataset on every iteration

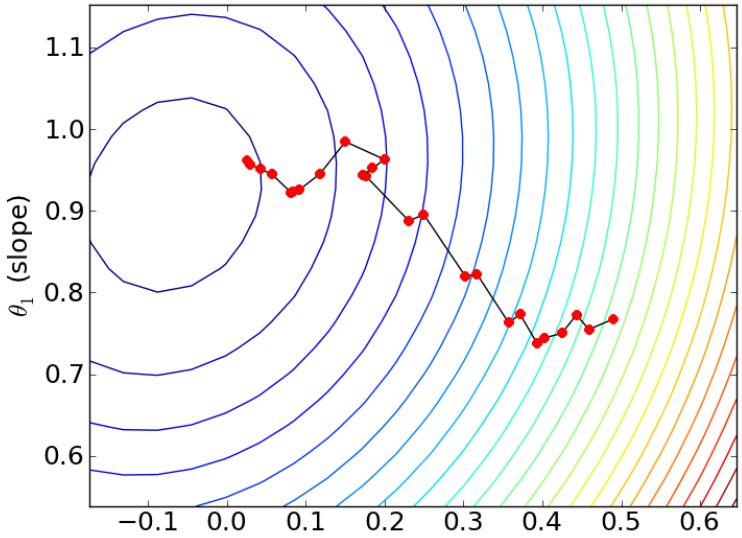
Stochastic Gradient Descent

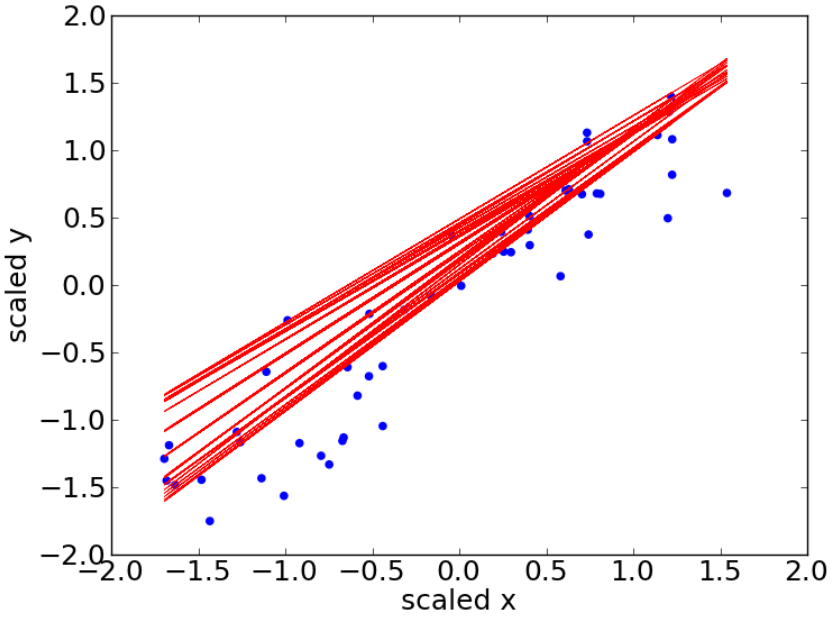
- At each step, pick one random data point
- Continue as if your entire dataset was just the one point

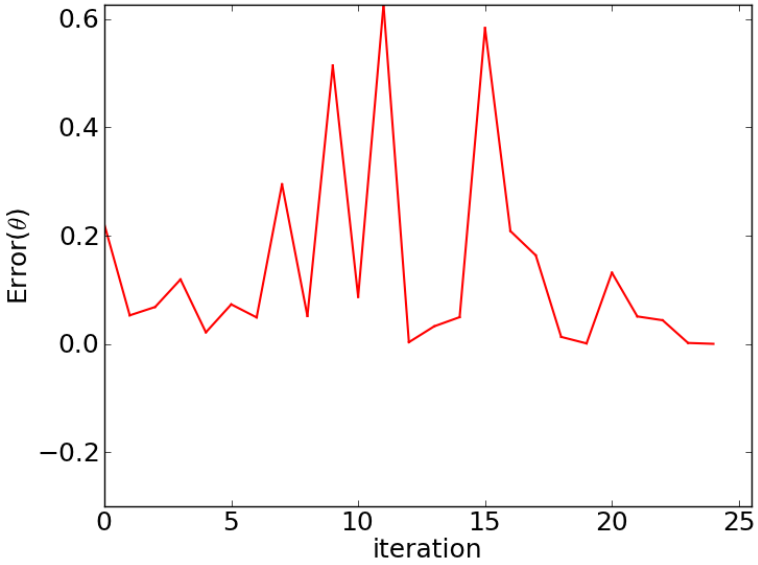
Minibatch Gradient Descent

- At each step, pick a small subset of data points
- Continue as if your entire dataset was just this subset

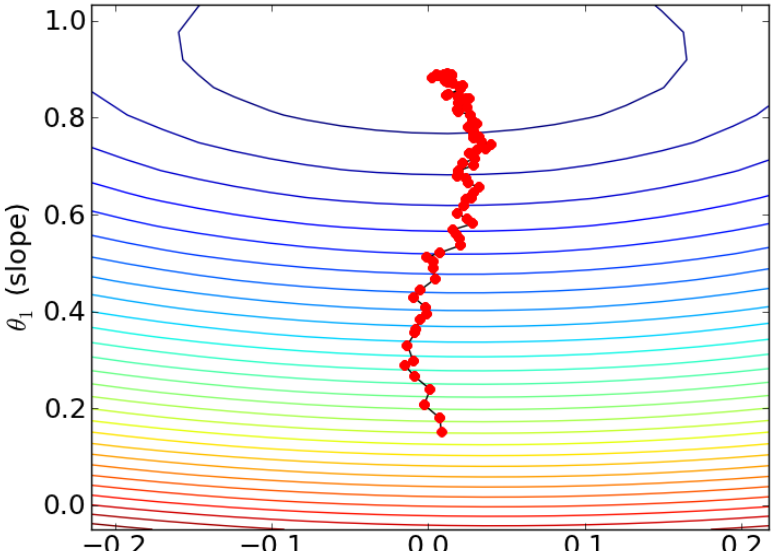
EXAMPLE USING STOCHASTIC GRADIENT DESCENT

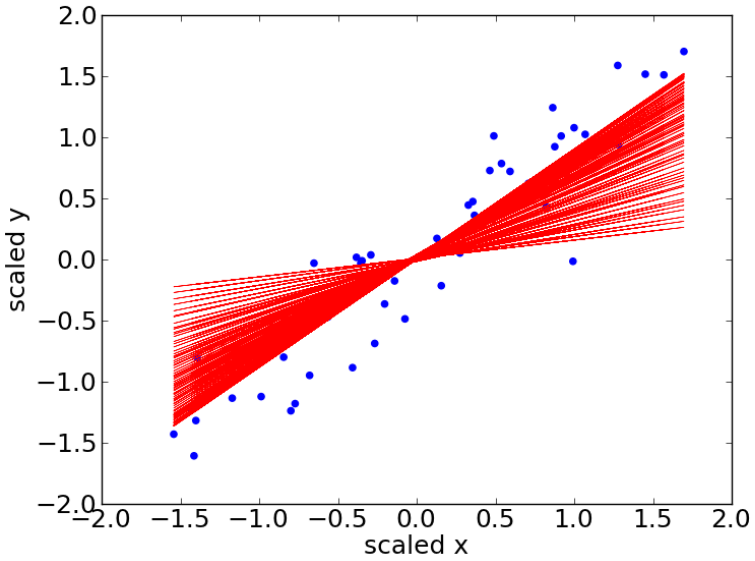


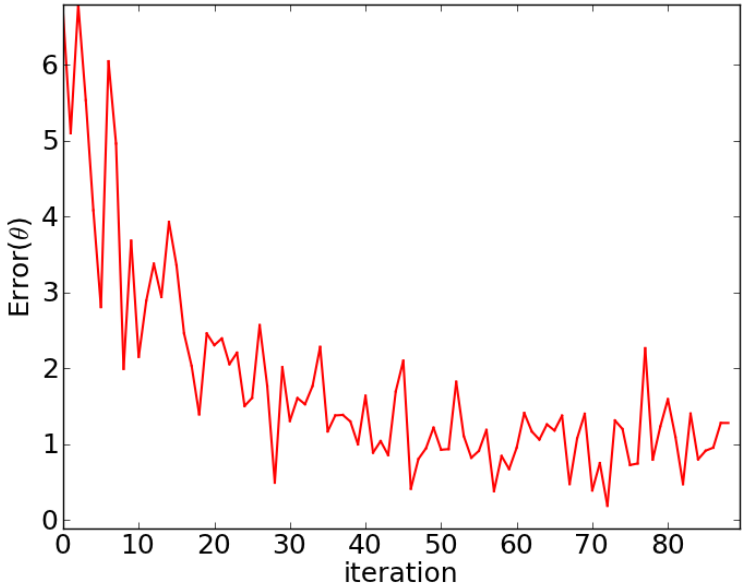




EXAMPLE USING MINIBATCHES







PARALLELIZING STOCHASTIC GRADIENT DESCENT

Stochastic Gradient Descent

- At each step, pick a random data point
- Continue as if your entire dataset was just the one point

Parallel Stochastic Gradient Descent

- In each of *k* threads, pick a random data point
- Compute the gradient and update the weights
- Weights will be "mixed"

HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent, Feng Niu, Benjamin Recht, Christopher Re, Stephen J. Wright, NIPS 2011

thread 0

$$\theta_0^{(1)} \leftarrow \theta_0^{(0)} + (\theta_0^{(0)} + \theta_1^{(0)} x_3 - y_3)$$

$$\theta_1^{(1)} \leftarrow \theta_1^{(0)} + (\theta_0^{(2)} + \theta_1^{(0)} x_3 - y_3) x_3$$

$$\theta_0^{(3)} \leftarrow \theta_0^{(2)} + (\theta_0^{(2)} + \theta_1^{(2)} x_5 - y_5)$$

$$\theta_1^{(3)} \leftarrow \theta_1^{(2)} + (\theta_0^{(4)} + \theta_1^{(2)} x_5 - y_5) x_5$$

thread 1

$$\theta_0^{(2)} \leftarrow \theta_0^{(1)} + (\theta_0^{(1)} + \theta_1^{(0)} x_8 - y_8)$$

$$\theta_1^{(2)} \leftarrow \theta_1^{(1)} + (\theta_0^{(2)} + \theta_1^{(1)} x_8 - y_8) x_8$$

$$\theta_0^{(4)} \leftarrow \theta_0^{(3)} + (\theta_0^{(3)} + \theta_1^{(2)} x_9 - y_9)$$

$$\theta_1^{(4)} \leftarrow \theta_1^{(3)} + (\theta_0^{(4)} + \theta_1^{(3)} x_9 - y_9) x_9$$

SLIDES CAN BE FOUND AT:

TEACHINGDATASCIENCE.ORG