Gradient Descent

Ott Toomet

May 4, 2017

Where We Stand

- Introduction
- Methods
  - python, pandas
  - Linear algebra
  - Probability and statistics
- Guest speaker
- Linear regression
  - Causality
- ML
  - ML Experiment design
  - Nearest neighbors
- Methods
  - Gradient Descent
  - Maximum Likelihood, logit
  - regularization
- ML
  - Naive bayes
  - PCA/dimensionality reduction
  - Clusters & recommenders
  - Trees and forests
  - Neural networks
- Wrap-up

Nearest Neighbors: Review

- Instance-based learning
- Lazy learning
- k-Nearest Neighbors
- Distance/metrics
- Curse of Dimensionality

Linear Regression

"Least Squares" means

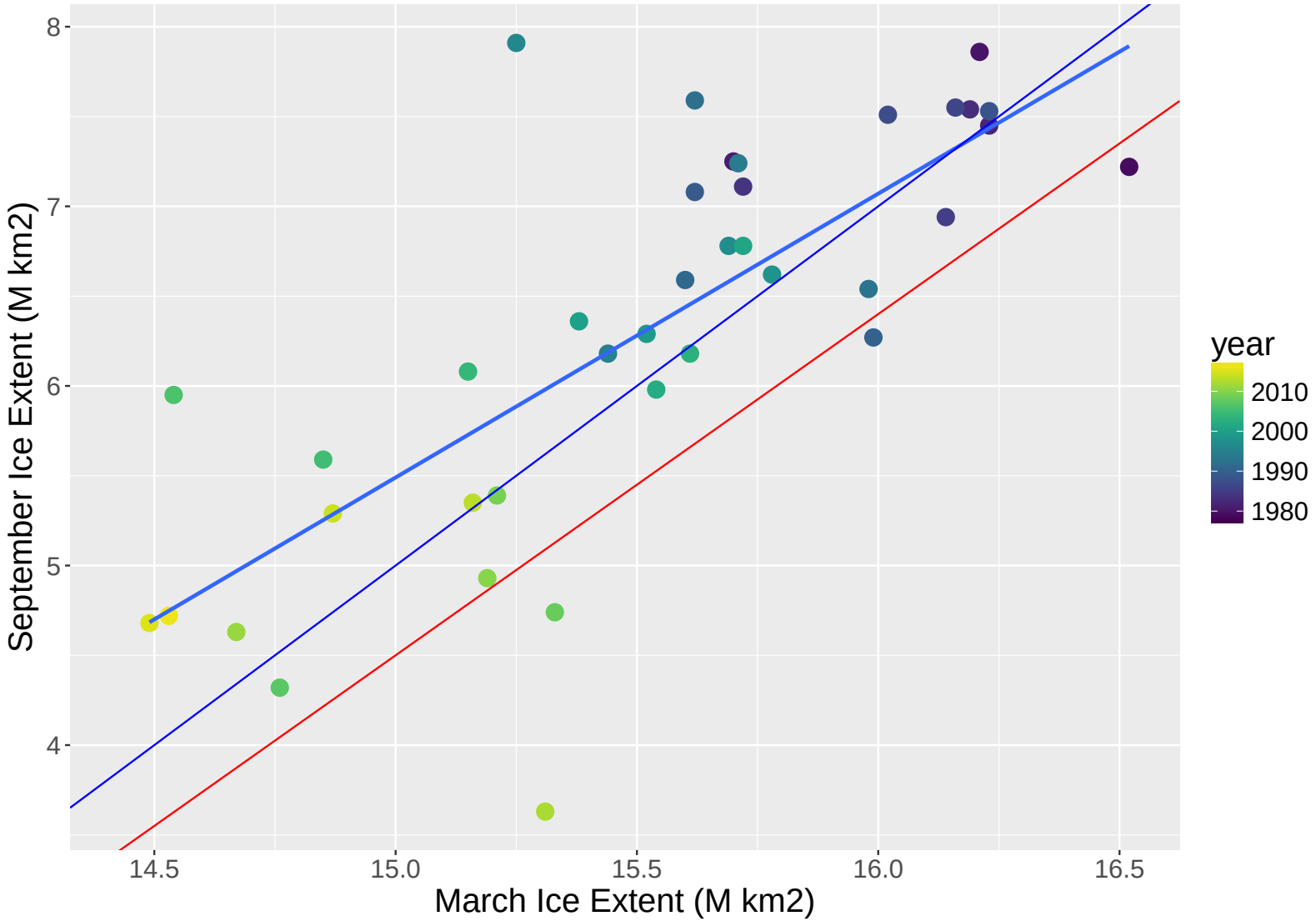
$$\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = \sum_i (\hat{y}_i - y_i)^2$$

where

$$\hat{y}_i = \hat{\boldsymbol{\beta}}' \boldsymbol{x}_i$$

- find $\boldsymbol{\beta}$ that minimizes $L(\boldsymbol{\beta})$
  - how?
  - L is "loss function" (objective function, cost function)

Example: Predict September Arctic Sea Ice by March Extent

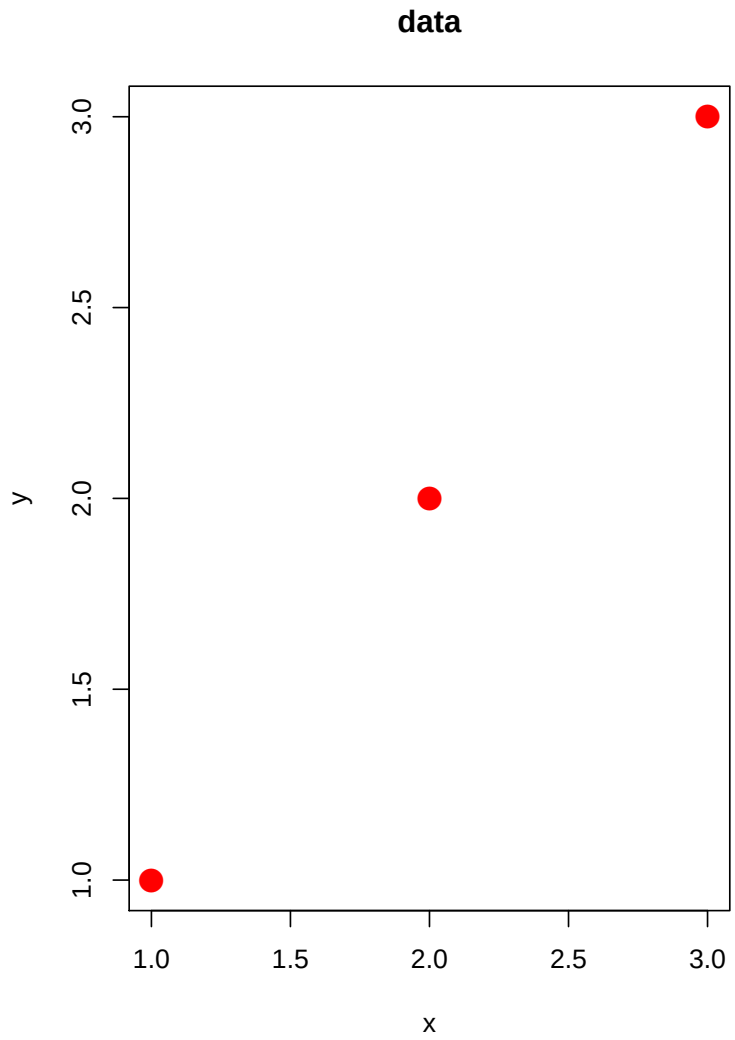


Try different β-s

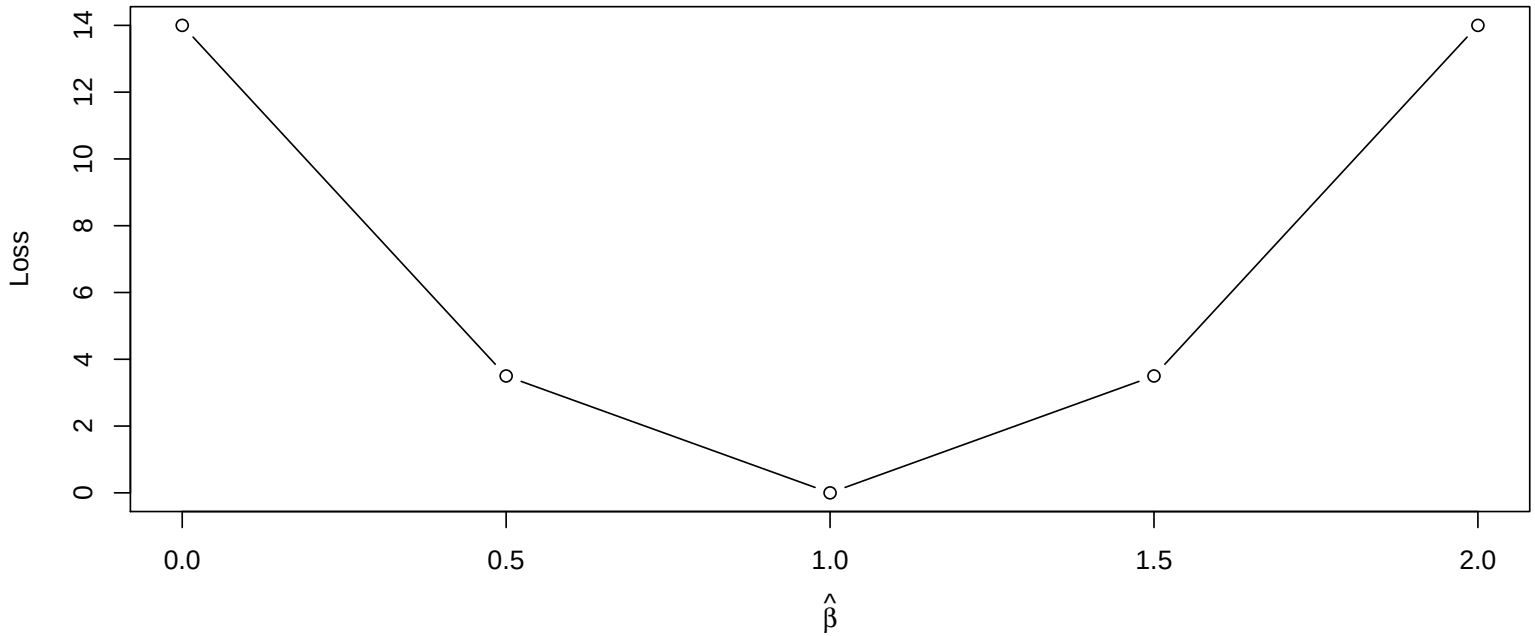
- $L(-25, 2) = NA$
- $L(-24, 1.9) = NA$
- $L(-18.217, 1.58) = 19.671$

Trial-and-Error Exercise (Grid Search)

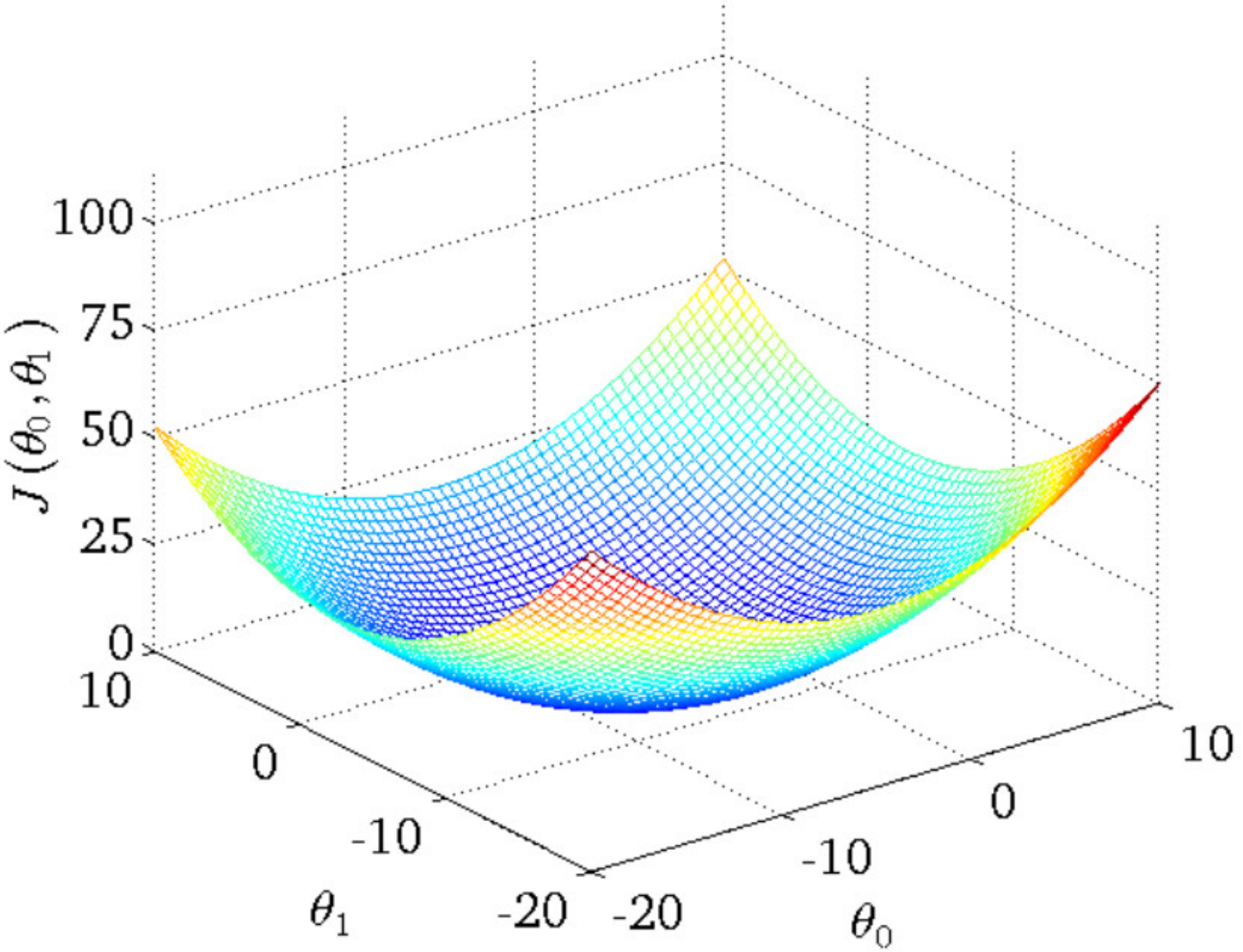
- OLS model $y_i = \beta x_i + \epsilon_i$
- Use the data at right
- Find the optimal β
  - Calculate $L(0)$, $L(0.5)$, $L(1)$, $L(1.5)$, $L(2)$.
- Plot $L(\beta)$ versus β



$L(\beta)$



2D Case



1 Review

2 OLS Example

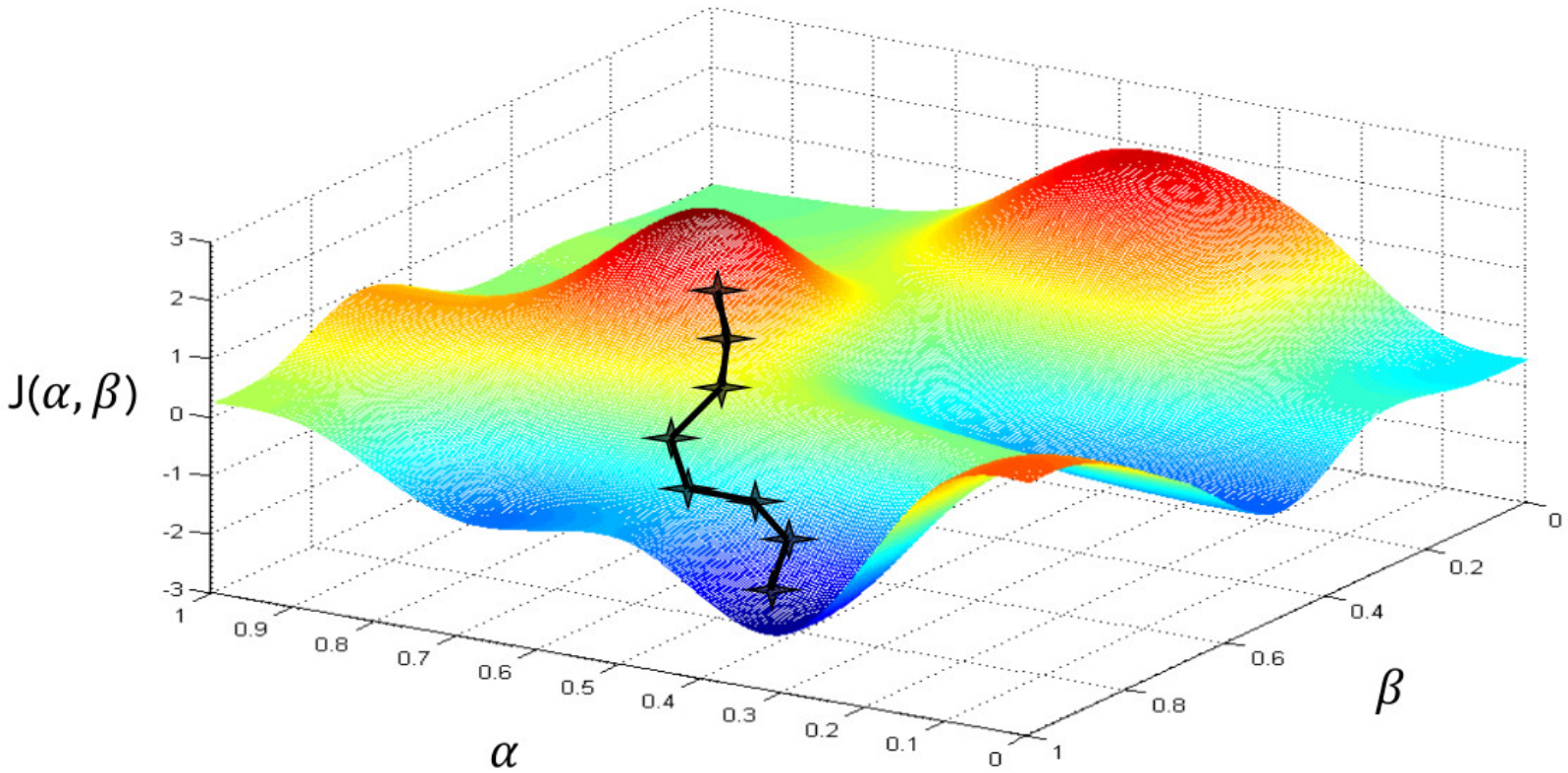
3 Gradient Descent
- What Is Gradient
- How to Optimize
- Problems
- Learning Rate
- Example
- Feature Scaling

4 Key Concepts

How To Minimize $L(\boldsymbol{\beta})$?

1. Start with a $\boldsymbol{\beta}$ value
2. Calculate $L(\boldsymbol{\beta})$.
3. Change $\boldsymbol{\beta}$ so it decresases L
4. Repeat 2-4 until we are at minimum

How To Descend



What is Gradient

Vector of first derivatives of the function with respect to it's arguments

- Direction where the function's growth is steepest
- "Speed" of growth
  - Per unit interval

Example:

$$f(\beta) = \beta^2 \quad \Rightarrow \quad \nabla f(\beta) = 2\beta$$

- positive if $\beta > 0$
  - $f(\cdot)$ grows when β grows
- negative if $\beta > 0$
  - $f(\cdot)$ grows when β decreases
- zero if $\beta = 0$
  - We are in a (local) optimum

Two-Dimensional Example

$$f(\beta_1, \beta_2) = \mathrm{e}^{-\beta_1^2 - \beta_2^2}$$

$$\frac{\partial f(\beta_1, \beta_2)}{\partial \beta_1} = -2 f(\beta_1, \beta_2) \beta_1$$

$$\frac{\partial f(\beta_1, \beta_2)}{\partial \beta_2} = -2 f(\beta_1, \beta_2) \beta_2$$

In vector form

$$\frac{\partial f(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}'} = -2 f(\boldsymbol{\beta}) \boldsymbol{\beta}$$

Linear Regression Example

Example of Linear Regression:

- In non-matrix form

$$L(\boldsymbol{\beta}) = \sum_i (y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2$$

$$\frac{\partial}{\partial \beta_1} L(\boldsymbol{\beta}) = 2 \sum_i (y_i - \beta_1 x_{1i} - \beta_2 x_{2i}) \cdot x_{1i}$$

$$\frac{\partial}{\partial \beta_2} L(\boldsymbol{\beta}) = 2 \sum_i (y_i - \beta_1 x_{1i} - \beta_2 x_{2i}) \cdot x_{2i}$$

- In matrix form

$$L(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{\beta}\mathsf{X})'(\boldsymbol{y} - \boldsymbol{\beta}\mathsf{X})$$

$$\frac{\partial}{\partial \boldsymbol{\beta}} L(\boldsymbol{\beta}) = 2(y_i - \boldsymbol{\beta}\mathsf{X})' X$$

How To Improve $\boldsymbol{\beta}$

Move in (the opposite) direction of *gradient* $\nabla f(\boldsymbol{\beta})$

- Gradient: in which direction the function grows most
- Climbing a snowy mountain in fog
- $-\nabla f(\boldsymbol{\beta})$: direction the function decreases most

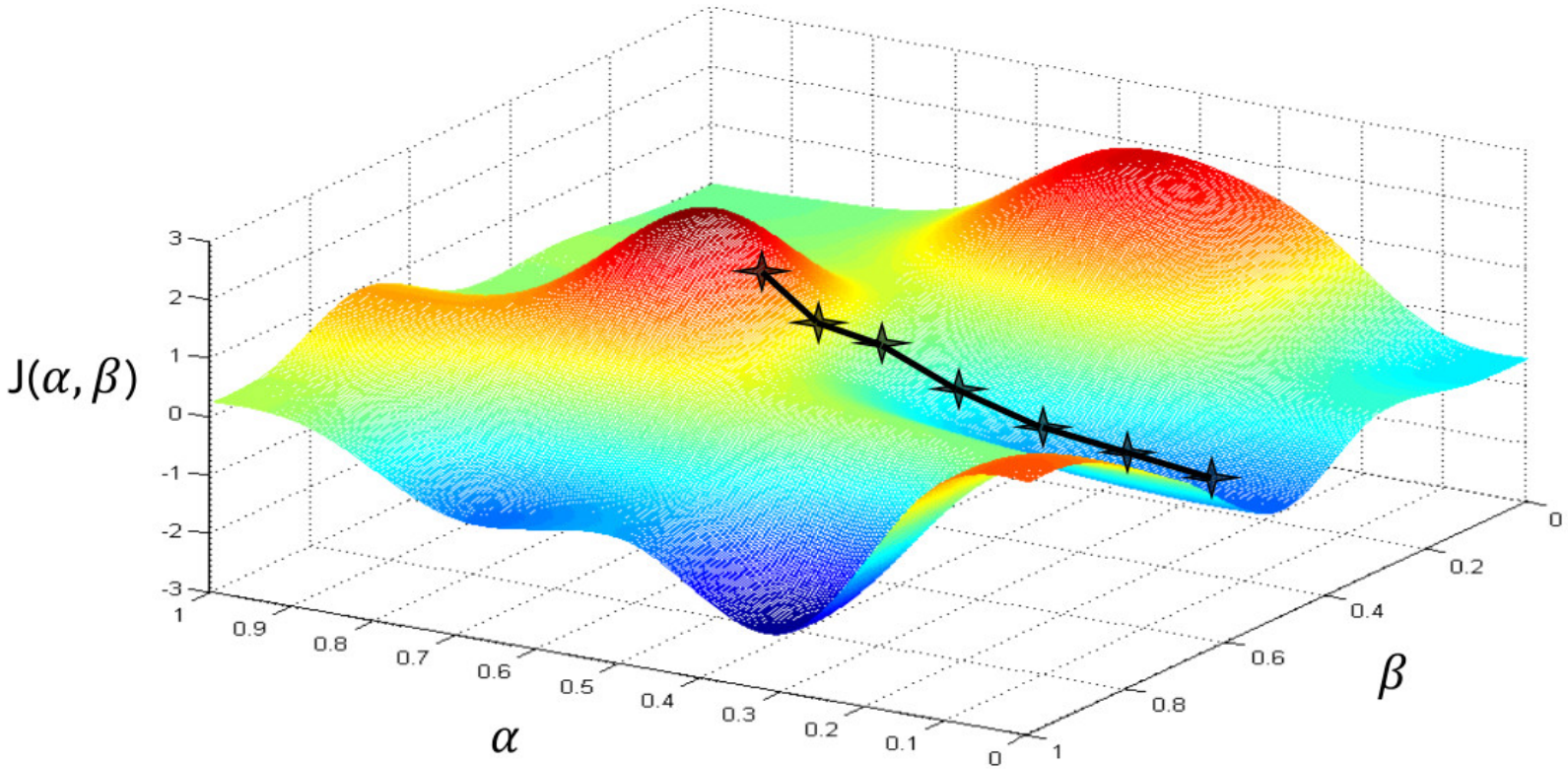
$$\boldsymbol{\beta}_{n+1} = \boldsymbol{\beta}_n - R\frac{\partial}{\partial \boldsymbol{\beta}'}L(\boldsymbol{\beta})$$

- R: step size (learning rate)
  - Should be small
  - Can be made adaptive
  - Can be calculated
- Stop when gradient close to zero ...
- or when the objective function does not decrease any more
- or when too many iterations

Algorithm

1. Set $n = 0$ and $\boldsymbol{\beta}^0$ to a value
   - $\boldsymbol{\beta}^{\mathbf{0}} = \mathbf{0}$ is sometimes a good choice
2. Choose R (a small number)
3. Calculate $L(\boldsymbol{\beta}^n)$
4. Calculate gradient $\nabla L(\boldsymbol{\beta}^n)$
5. Is gradient close to $\mathbf{0}$?
   - Yes – stop
6. Calculate $\boldsymbol{\beta}^{n+1} = \boldsymbol{\beta}^n - R \cdot \nabla L(\boldsymbol{\beta})$
7. Calculate $L(\boldsymbol{\beta}^{n+1})$
8. Did $L(\boldsymbol{\beta})$ decrease substantially?
   - No – stop
9. Is n too large?
   - Yes – stop
10. set $n := n + 1$, repeat 4

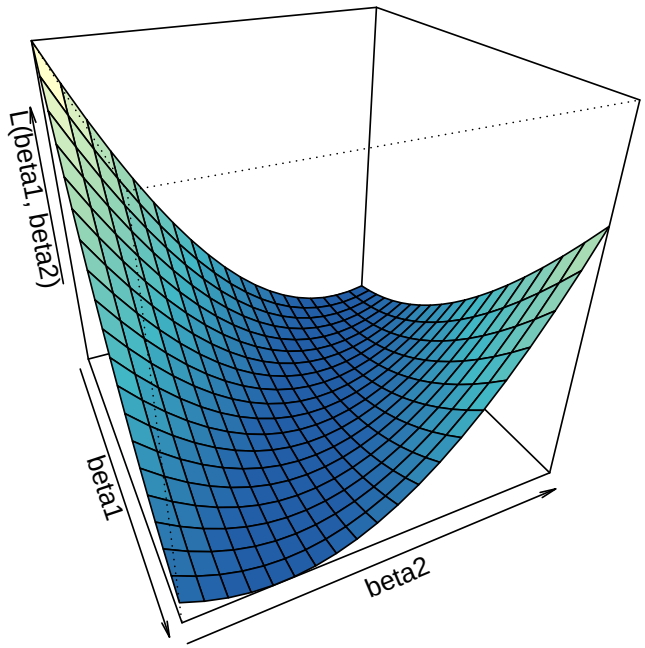
Local Minimum



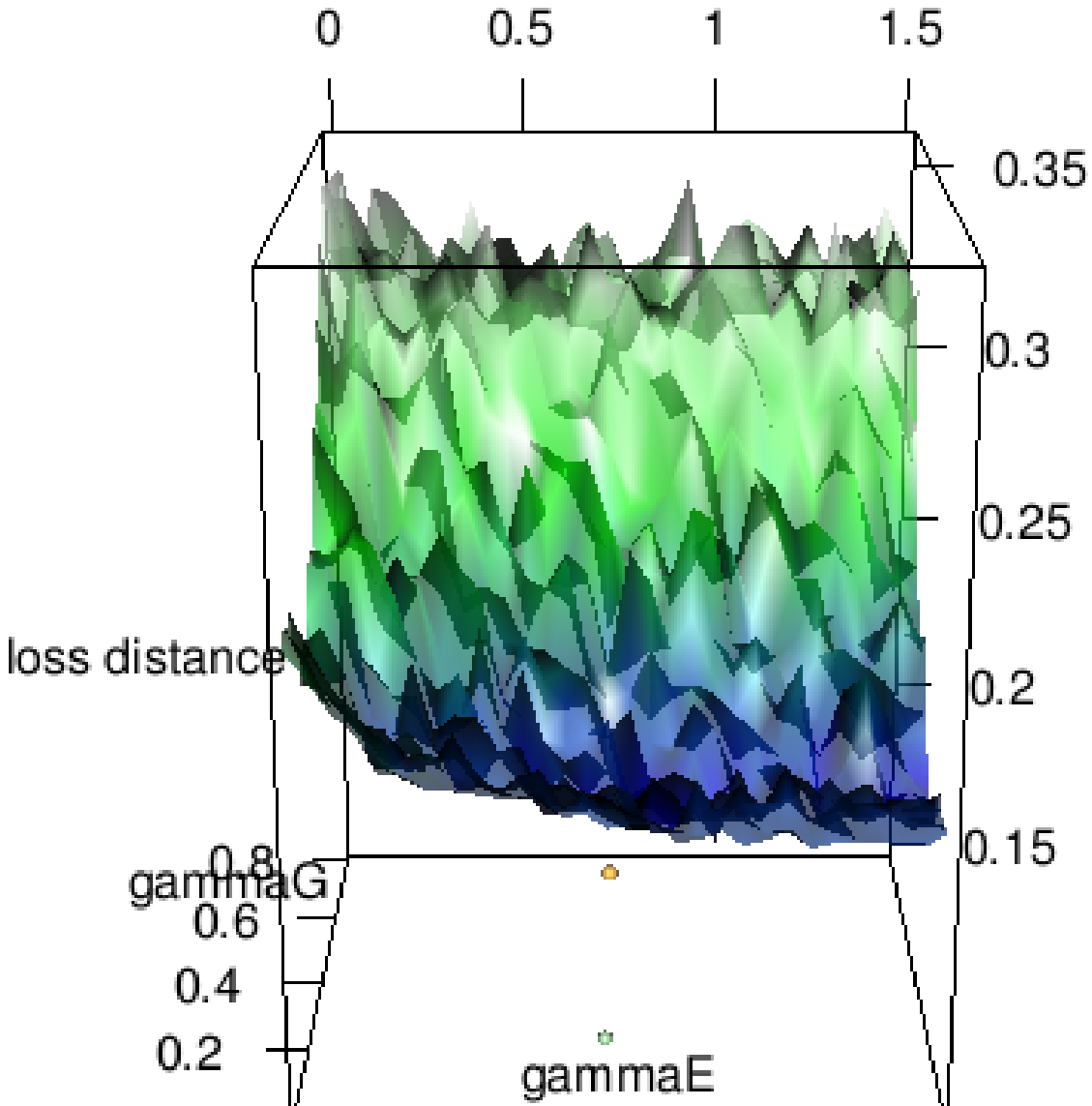
Convexity

Function $f(\boldsymbol{x})$ is convex iff:

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2, \quad t \in (0,1)$$
$$f(t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2) <$$
$$< tf(\boldsymbol{x}_1) + (1-t)f(\boldsymbol{x}_2)$$

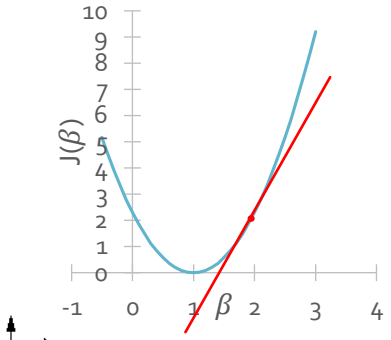


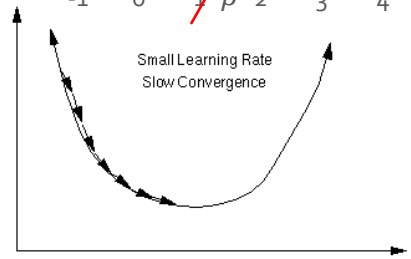
Noisy Objective Function

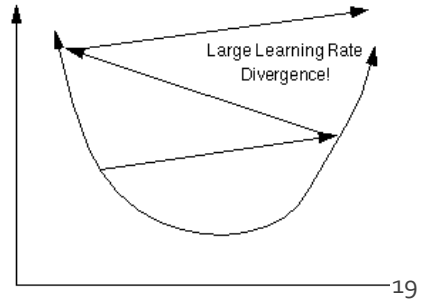


Gradient Descent: Learning Rate

- β <- β - R$\frac{\partial}{\partial \beta} J(\alpha, \beta)$
- What does R do?
- Small R:
  - Gradient descent can be slow
- Large R:
  - Can overshoot the minimum
  - May fail to converge
  - May diverge!







Gradient Descent: Convergence

- Do we need to change the learning rate?

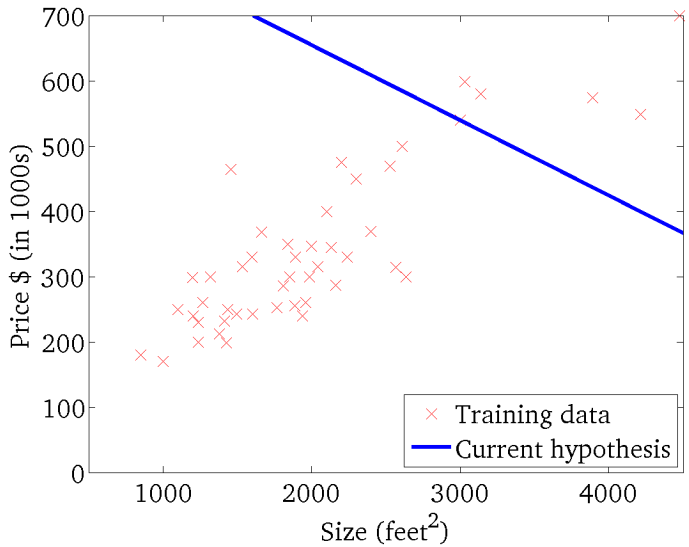
```
Choose an initial vector of parameters α, β
Choose learning rate R
Repeat until an approximate minimum is obtained:
```

$$\alpha \leftarrow \alpha - R\frac{\partial}{\partial \alpha}J(\alpha,\beta)$$

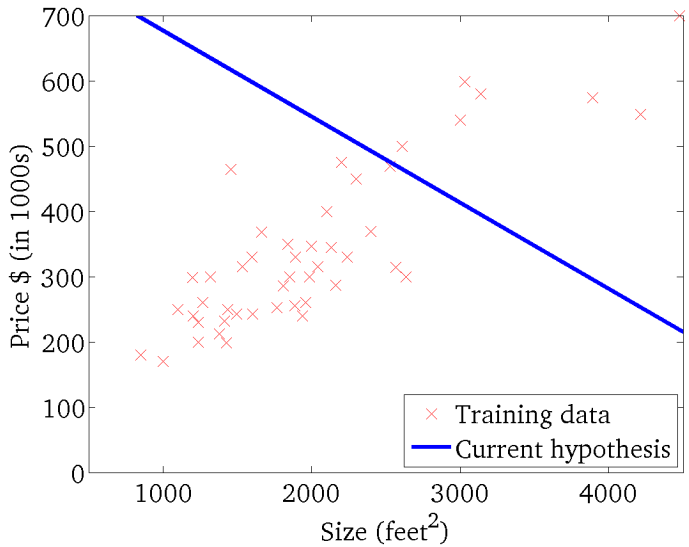
$$\beta \leftarrow \beta - R\frac{\partial}{\partial \beta}J(\alpha,\beta)$$

- No! Gradient descent can converge to a local minimum, even with the learning rate fixed
  - As we approach a local minimum, gradient descent takes smaller steps

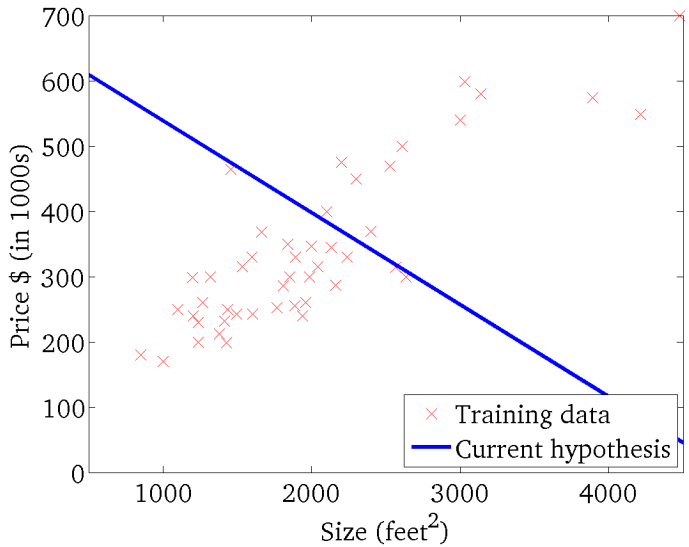
Gradient Descent: In Action



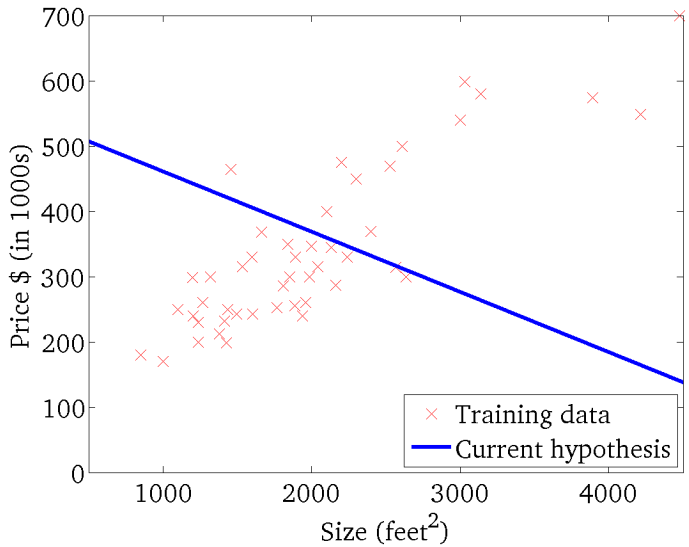
Gradient Descent: In Action



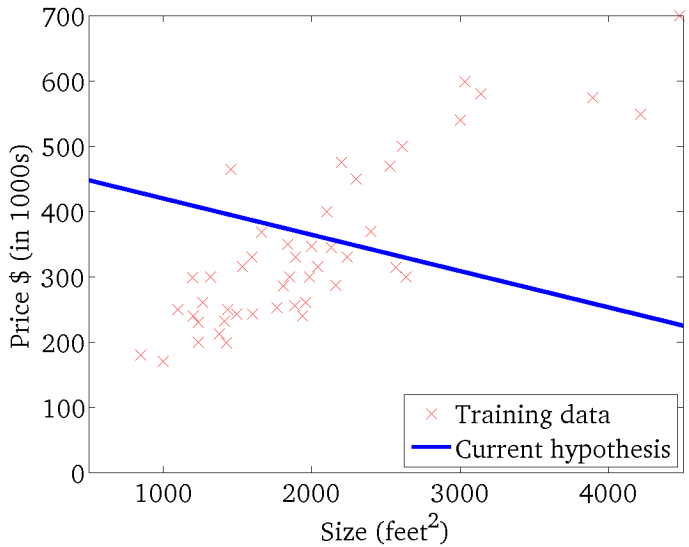
Gradient Descent: In Action



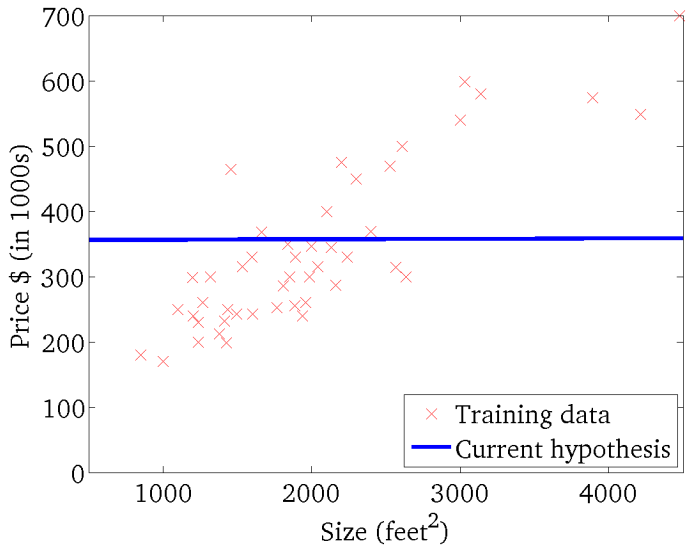
Gradient Descent: In Action



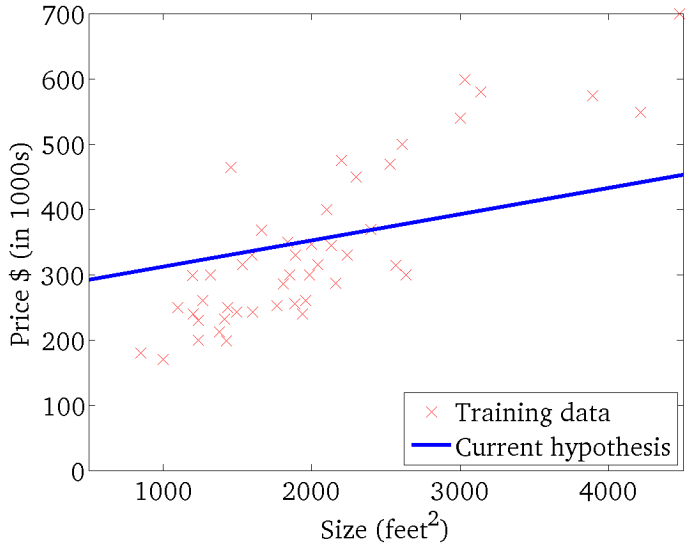
Gradient Descent: In Action



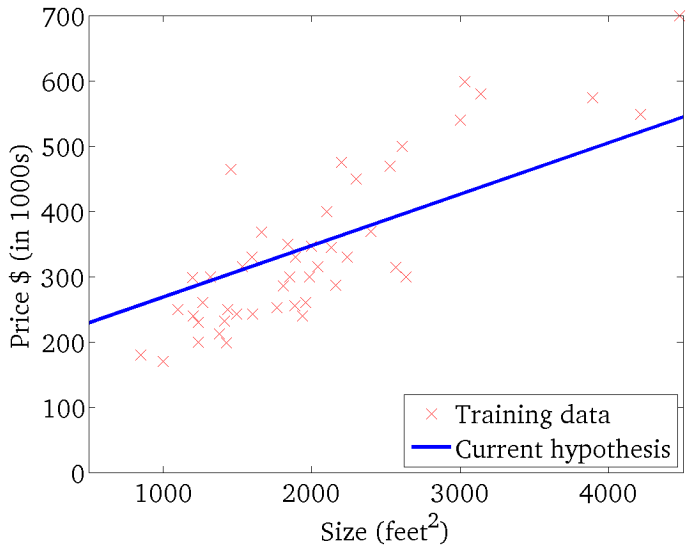
Gradient Descent: In Action



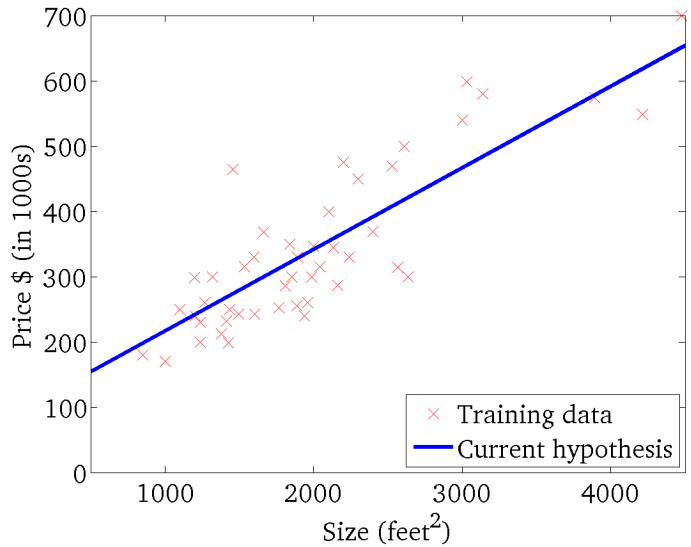
Gradient Descent: In Action



Gradient Descent: In Action



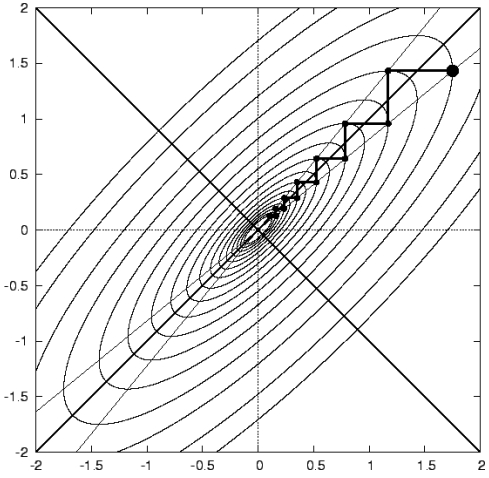
Gradient Descent: In Action



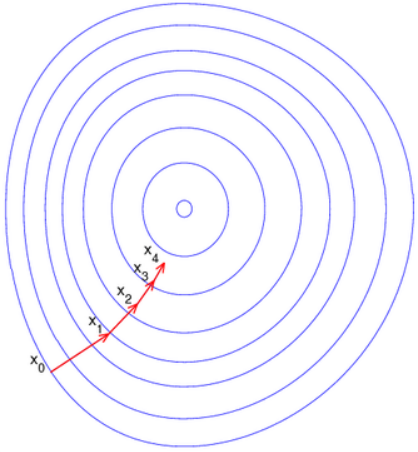
Feature scaling

- When some features (axes) are on different scales, gradient descent can be inefficient
- Putting different features on same scale can make gradient descent much faster

Not Proportional



Proportional



Key Concepts

- Cost Function (Loss Function)
- Non-Linear Optimization
- Gradient Descent
- Local/Global minima
- Convexity
- Learning Rate
- Feature Scaling