# Real Data and Vector Spaces

- What about data with more complex schemas?

| $x_1$ | $x_2$ | Date | prod_id | comment | y |
|---|---|---|---|---|---|
| 1.1 | 2.7 | 8/21/16 | 7 | "the best glider …" | 3.6 |
| 4.2 | 3.2 | 8/14/16 | 3 | "vacation for two …" | 7.5 |
| 9.8 | 9.2 | 9/20/16 | 4 | "A special gift for …" | 17 |
| … | … | … | … | … | … |

  - The math wants the features to be vectors …
- How do we encode dates, categorical fields, and text?

# Encoding Categorical Data

- How do we represent fields like "Product Category"
- **Proposal 1:** *Enumerate categories*
  - Sports = 1, Furniture = 2, Clothing = 3, Shoes = 4, …
  - Store field number as a feature
  - **Implications**:
    - **similarity:** sports is closer to Furniture than shoes
    - **magnitude:** larger values ➔ ?
  - Not typically used (unless there are two categories ...)

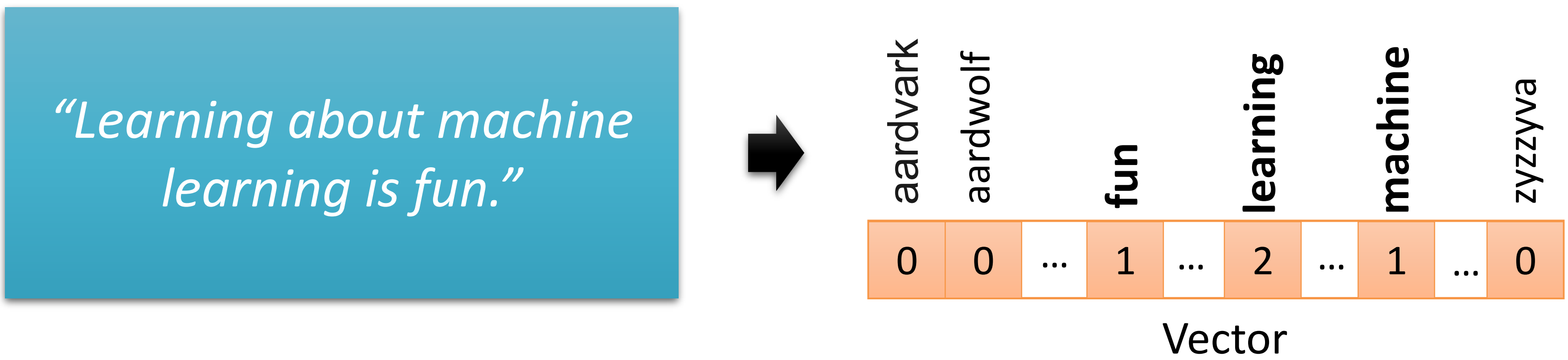
# One-hot encoding

- ➢ How do we represent fields like “Product Category”
- ➢ **Proposal 1:** *Enumerate categories*
  - Not typically used (unless there are two categories ...)
- ➢ **Proposal 2:** *Encode as binary vectors:*
  - Very commonly used and built-in to many packages
  - Enumerate all possible product categories (*m*)
  - Add *m* additional features to the record:
  - Put a one in the feature corresponding to the product category and a zero everywhere else.

| Sports | Furniture | Clothing | Shoes | Electronics | Music | Books |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |

# Working with Text Data

➢How do we convert text to vectors?




➢Bag-of-words model

- Transform emails into *d*-dimensional vectors
  - *d* is the number of unique words in the language (big!)
- Each entry is number of occurrences of that word
- **Sparse**: Most words don't occur in most emails
- **Remove Stop-Words:** common words that provide little information (e.g., "is", "about")

# The Linear Model

Data:

| $x_1$ | $x_2$ | y |
|---|---|---|
| 1.1 | 2.7 | 3.6 |
| 4.2 | 3.2 | 7.5 |
| 9.8 | 9.2 | 17 |
| … | … | … |

$$f_\theta(x) := \overbrace{\theta}^{\text{Vector of Parameters}}{}^T \overbrace{x}^{\text{Vector of Features}}$$

- ➢Encode data is real valued vectors
- ➢**Next:** find the optimal value for $\theta$
  - How?

$$\theta, x \in \mathbb{R}^p$$

# Finding the Best Parameters

Model: $$f_\theta(x) := \theta^T x$$

**Step 1:** define a **Loss Function:**

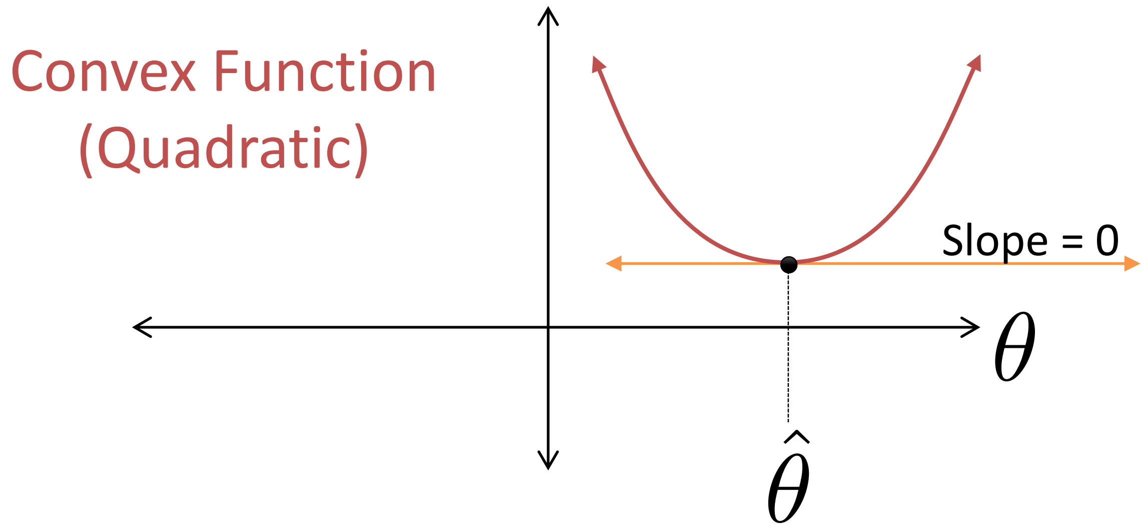
$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_\theta(x_i)\right)^2$$

**Step 2:** Search for best model parameters $\theta$

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_\theta(x_i)\right)^2$$

# Minimizing the Squared Error

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2$$




- ➢Take the gradient and set it equal to zero

# Minimizing the Squared Error

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2$$

- Setting equal to zero and solving for $\theta$:

$$\sum_{i=1}^{n} (\theta^T x_i) x_i = \sum_{i=1}^{n} y_i x_i \Rightarrow X^T X \theta = X^T y$$

- Normal Equation:

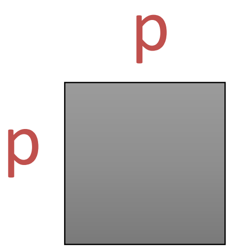
$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

- Solved using any standard linear algebra library

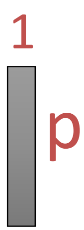
# Least Squares Regression using the **Statistical Query Pattern**

$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

- In database compute sums:




$$C = X^T X = \sum_{i=1}^{n} x_i x_i^T \qquad O(np^2)$$




$$d = X^T y = \sum_{i=1}^{n} x_i y_i \qquad O(np)$$

- On client compute:

$$\hat{\theta} = C^{-1} d \qquad O(p^3)$$

- Rather than directly solving:

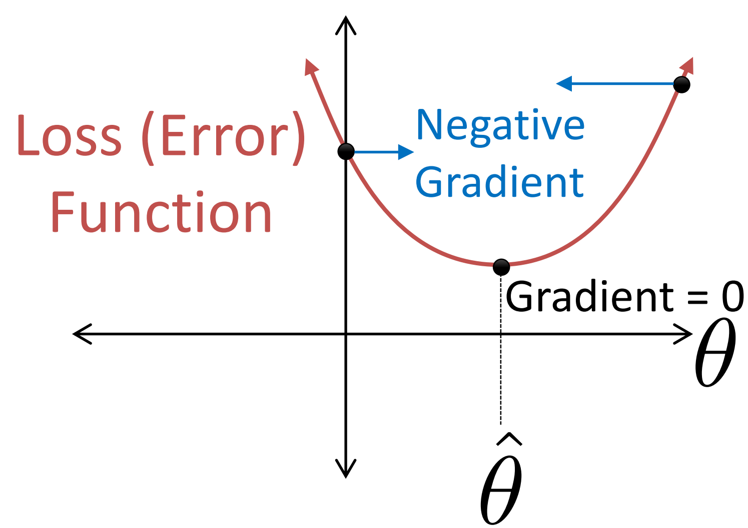
$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \theta^T x_i)$$

- Instead we compute the gradient of the loss:

$$G(\theta; X, y) = \nabla_\theta \frac{1}{n} \sum_{i=1}^{n} L(y_i, \theta^T x_i) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta L(y_i, \theta^T x_i)$$

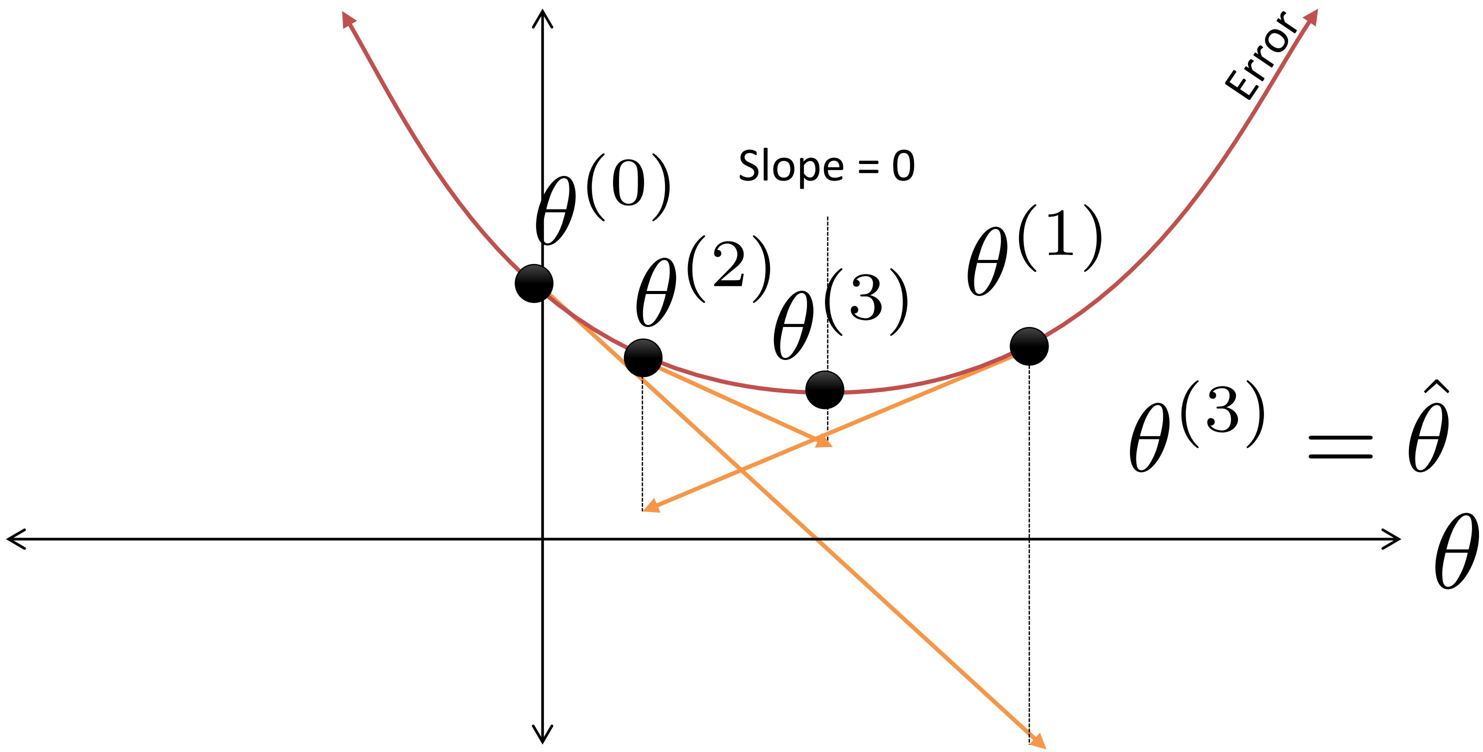
$$= \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{y_i} - \theta^T x_i) x_i$$

- **Big Idea:** Negative gradient points in the direction of steepest descent



# **Gradient Descent** Algorithm

```
t ← 0
θ^(0) ← Vec(0)
while (not converged):
    θ^(t+1) ← θ^(t) – Stepsize^(t) * G(θ; X,Y)
    t ← t + 1
```



# **Gradient Descent** Algorithm

$t \leftarrow 0$

$\theta^{(0)} \leftarrow \mathrm{Vec}(0)$

while (not converged):

  $\theta^{(t+1)} \leftarrow \theta^{(t)} - \mathrm{Stepsize}^{(t)} * G(\theta; X, y)$

  $t \leftarrow t + 1$

- Does this fit the statistical query pattern
  - Yes! Only dependence on data is:

Size? 1 p

$$G(\theta; X, y) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)x_i$$

Complexity? $O(np)$

  - Can we go even faster?
    - **Stochastic Gradient Descent (SGD)**: Approximate the gradient by sampling data (typically several hundred records per query).

# Stochastic Gradient Descent

- Update the parameters for each training case in turn, according to its own gradients

```
t ← 0
θ^(0) ← Vec(0)
while (not converged):
    θ^(t+1) ← θ^(t) – Stepsize^(t) * G(θ; X,y)
    t ← t + 1
```

$$G(\theta; X, y) = (y_i - \theta^T x_i) x_i$$

$$\mathrm{Stepsize}^{(\mathrm{t})} = \frac{1}{t+1}$$

Complexity?

$O(p)$

# Bias-Variance Tradeoff

- So far we have minimized the **training error**: the error on the training data.
  - low training error does not guarantee good expected performance (due to over-fitting)
- We would like to reason about the **test error**

**Theorem:**

Test Error = Noise + Bias$^2$ + Variance

**Noisy data** has inherent error (measurement error)

Error due to models **inability to fit** the data. **(Under Fitting)**

Error due to inability to estimate model parameters. **(Over-fitting)**

# Bias Variance Plot

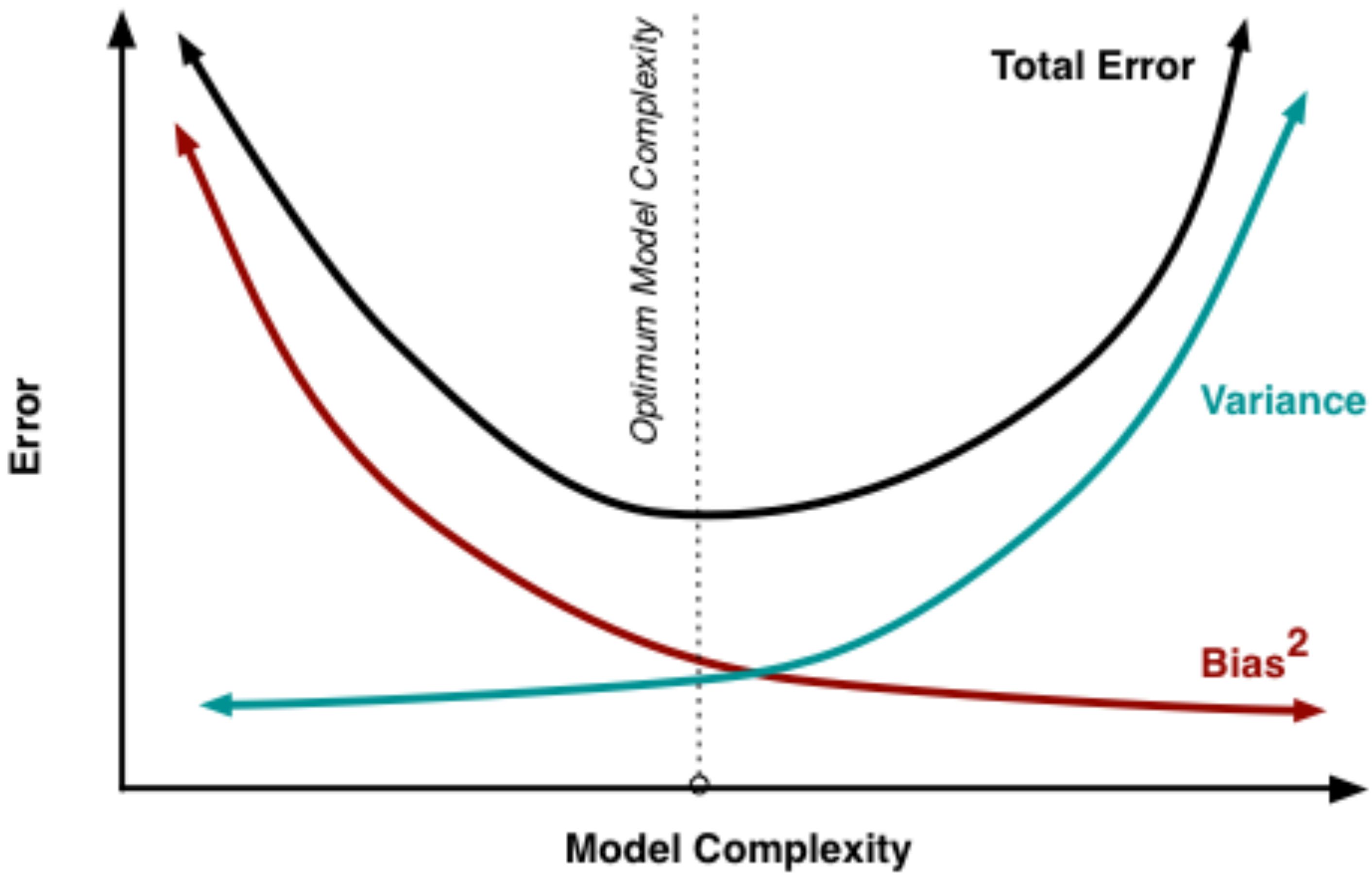



Image from http://scott.fortmann-roe.com/docs/BiasVariance.html

# Regularization to Reduce Over-fitting

- High dimensional models tend to over-fit
  - How could we “favor” lower dimensional models?
- **Solution Intuition:**
  - Too many features → over-fitting

$$f(x) = \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_d x_d$$

  - Force many of the $\theta_i \approx 0$ (e.g., $i > 2$) (“effectively fewer features”)

$$\begin{aligned} f(x) &= \theta_1 x_1 + \theta_2 x_2 + 0 x_3 + \ldots + 0 x_d \\ &= \theta_1 x_1 + \theta_2 x_2 \end{aligned}$$

  - Keeping weights close to zero **reduces variance**

# Regularization to Reduce Over-fitting

- We can add a regularization term:

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \overbrace{(y_i - \theta^T x_i)^2}^{\text{Squared Loss}} + \underbrace{\lambda}_{\text{Regularization Parameter}} \overbrace{R(\theta)}^{\text{Regularization Function}}$$

- Common of Regularization Functions:

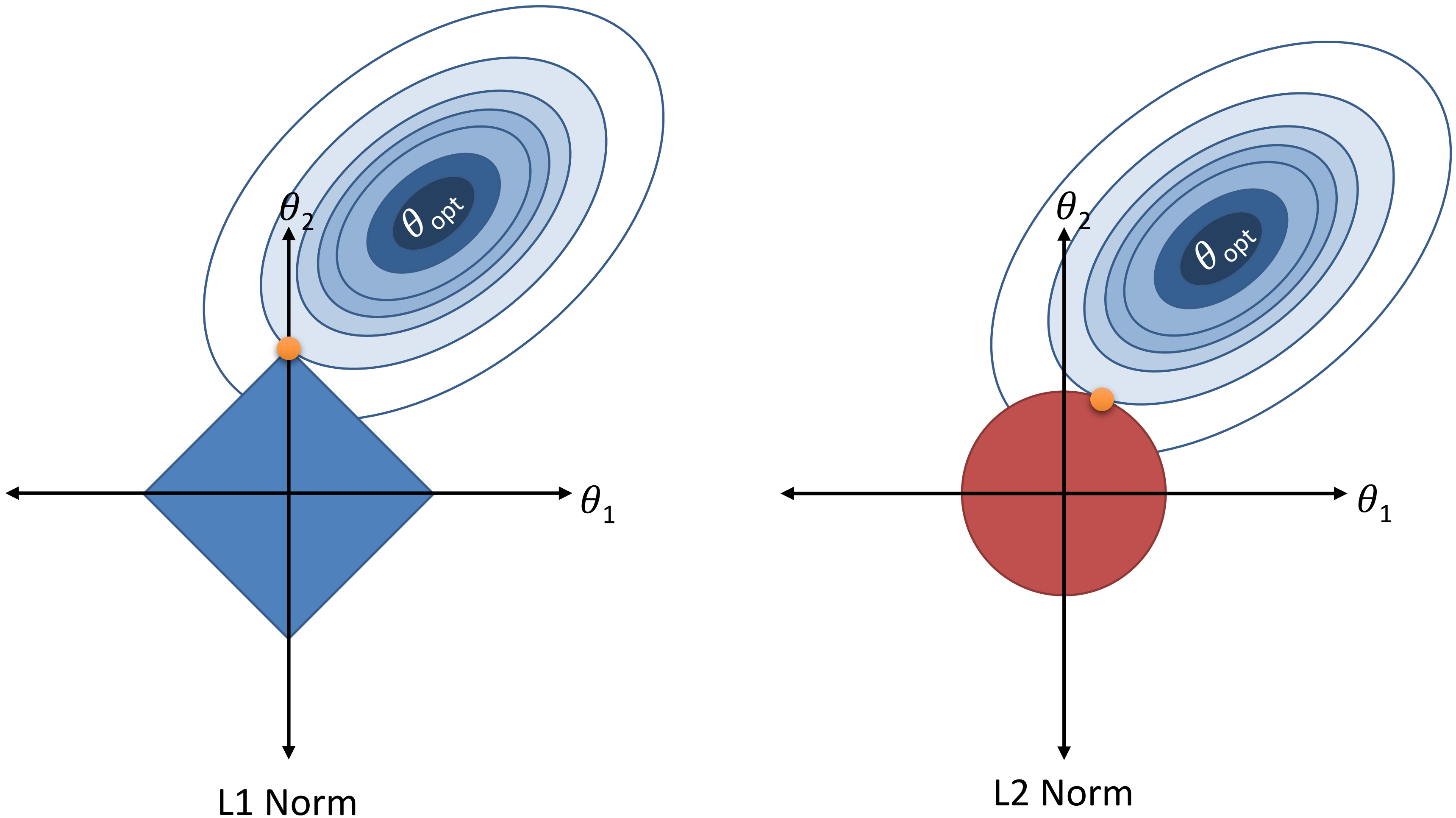
Ridge (L2-Reg) Regression $R_{\text{Ridge}}(\theta) = \sum_{i=1}^{d} \theta_i^2$

Lasso (L1-Reg) $R_{\text{Lasso}}(\theta) = \sum_{i=1}^{d} |\theta_i|$

  - Encourage small parameter values

- The parameter λ determines amount of reg.
  - Larger → more reg. → lower variance → higher bias

# Regularization and Norm Balls



# Regularization to Reduce Over-fitting

- We can add a regularization term:

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \quad \frac{1}{n} \sum_{i=1}^{n} \overbrace{(y_i - \theta^T x_i)^2}^{\text{Squared Loss}} + \underbrace{\lambda}_{\substack{\text{Regularization}\\ \text{Parameter}}} \overbrace{R(\theta)}^{\substack{\text{Regularization}\\ \text{Function}}}$$

- Solving the regularized problem:
  - Closed form solution for Ridge regression (L2):

$$\hat{\theta}_{\text{Ridge}} = (X^T X + \lambda I)^{-1} X^T Y$$

  - Iterative methods for Lasso (L1):
    - Stochastic gradient …
- How do we choose λ?

# Picking The Regularization Parameter λ

- **Proposal:** Minimize **training** error

$$\arg \min_{\theta \in \mathbb{R}^p, \boxed{\lambda \geq 0}} \quad \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2 + \lambda R(\theta)$$

  - Trivial solution ➔ $\lambda = 0$
- Intuition we want to minimize **test** error
  - **Test error**: error on unseen data
- **2nd Proposal:** Split training data into training and evaluation sets
  - For a range of λ values compute optimal $\theta_\lambda$ using only the reduced training set
  - Evaluate $\theta_\lambda$ on the separate evaluation set and select the λ with the lowest error

# Bias Variance Plot

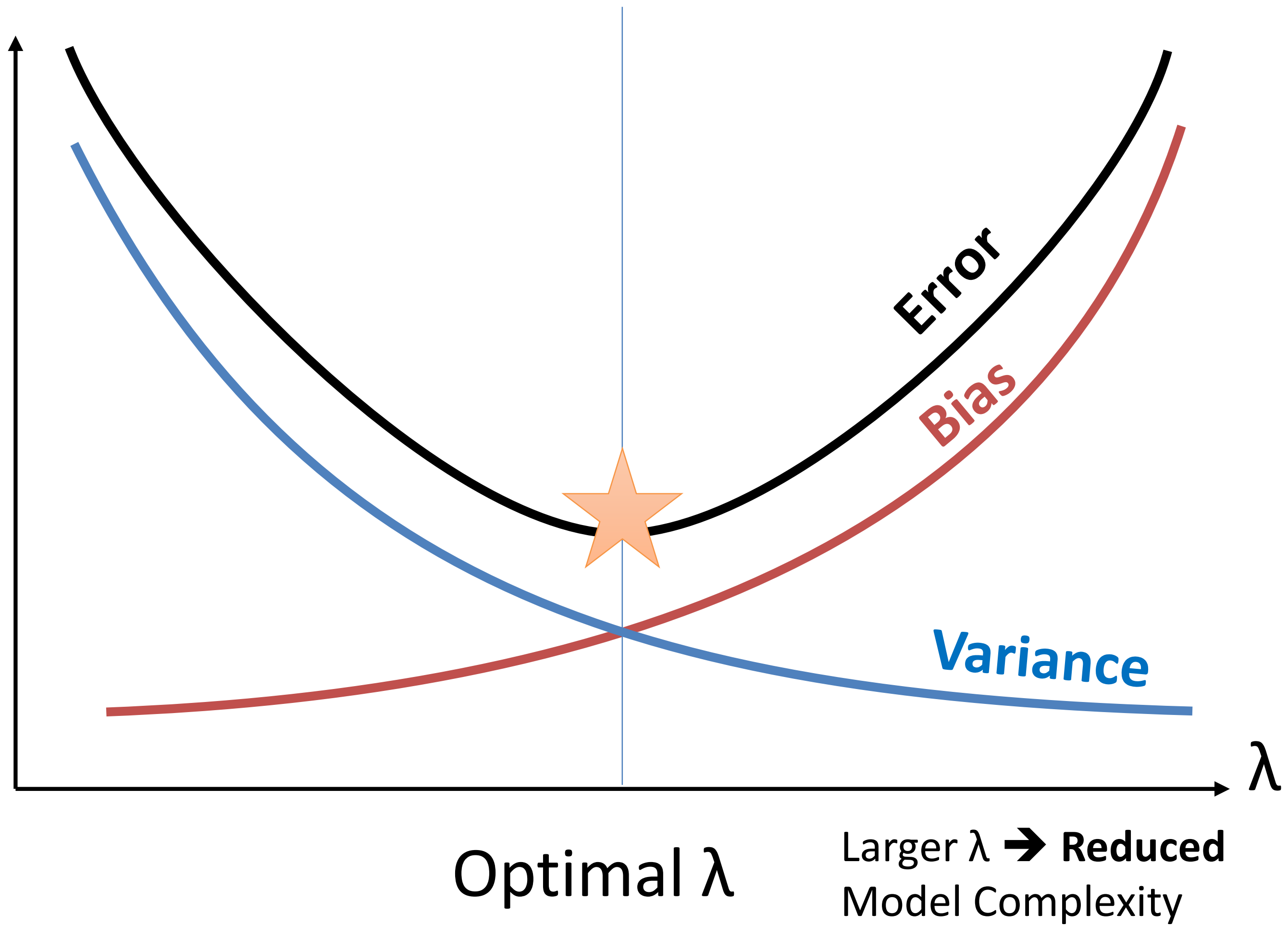



Image from http://scott.fortmann-roe.com/docs/BiasVariance.html