Announcements

- Homework 1 due **today (Wednesday) at 8pm**
- Homework 2 will be released tonight
  - Covers linear regression
- Quiz 1 released tomorrow (Thursday) at 8pm

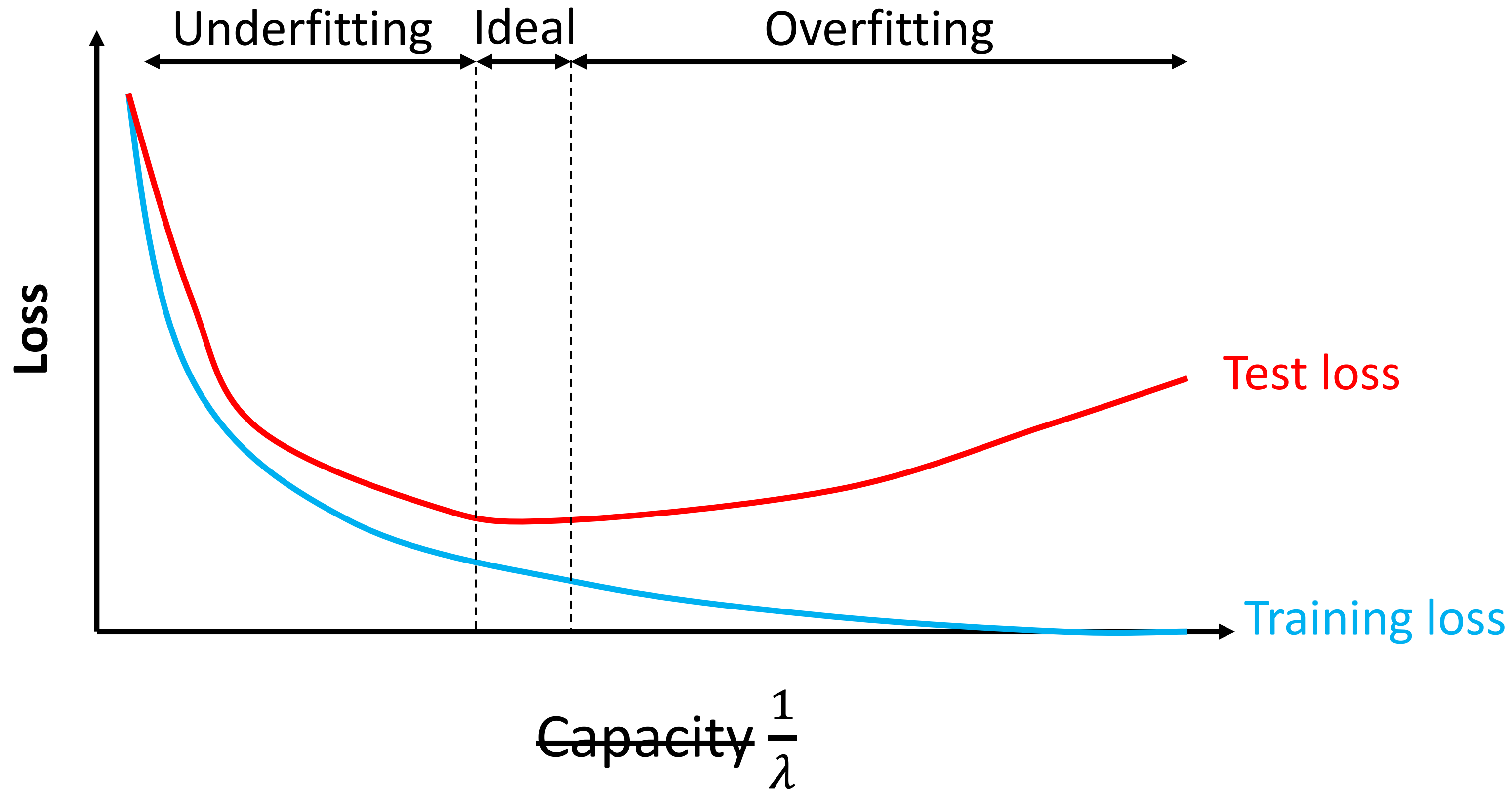
Recap: L_2 Regularization

- **Original MSE loss** + **regularization:**

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \cdot \|\beta\|_2^2$$

- λ is a **hyperparameter** that must be tuned (satisfies $\lambda \geq 0$)

Recap: L_2 Regularization



Recap: Cross Validation

- **Original MSE loss + regularization:**

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \cdot \|\beta\|_2^2$$

- λ is a **hyperparameter** that must be tuned (satisfies $\lambda \geq 0$)

- How to choose λ?

Recap: Cross Validation

Training data Z_{train} | Val data Z_{val} | Test data Z_{test}

$\lambda_1 = 0.01 \qquad \hat{\beta}_1 \leftarrow \hat{\beta}(Z_{\text{train}}, \lambda_1) \qquad L_{\text{val}}^1 \leftarrow L\left(\hat{\beta}_1; Z_{\text{val}}\right)$

$\lambda_2 = 0.10 \qquad \hat{\beta}_2 \leftarrow \hat{\beta}(Z_{\text{train}}, \lambda_2) \qquad L_{\text{val}}^2 \leftarrow L\left(\hat{\beta}_2; Z_{\text{val}}\right) \qquad L\left(\hat{\beta}_{t'}; Z_{\text{test}}\right)$

$\lambda_2 = 1.00 \qquad \hat{\beta}_3 \leftarrow \hat{\beta}(Z_{\text{train}}, \lambda_3) \qquad L_{\text{val}}^3 \leftarrow L\left(\hat{\beta}_3; Z_{\text{val}}\right)$

$$t' \leftarrow \arg\min_t L_{\text{val}}^t$$

Recap: Cross Validation

- Generally important for tuning design choices
  - Hyperparameters
  - Features in the feature map
  - Model family
  - …

- Alternative approaches exist for very small datasets
  - Re-train on $Z_{\text{train}} \cup Z_{\text{val}}$
  - k-fold cross validation

Lecture 3: Linear Regression (Part 3)

CIS 4190/5190

Fall 2022

Agenda

- **Minimizing the MSE Loss**
  - Closed-form solution
  - Gradient descent

Recap: Linear Regression in Matrix Form

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \begin{bmatrix} \beta^\top x_1 \\ \vdots \\ \beta^\top x_n \end{bmatrix}$$

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 = \frac{1}{n}\|Y - X\beta\|_2^2$$

$$\|z\|_2^2 = \sum_{i=1}^{n} z_i^2$$

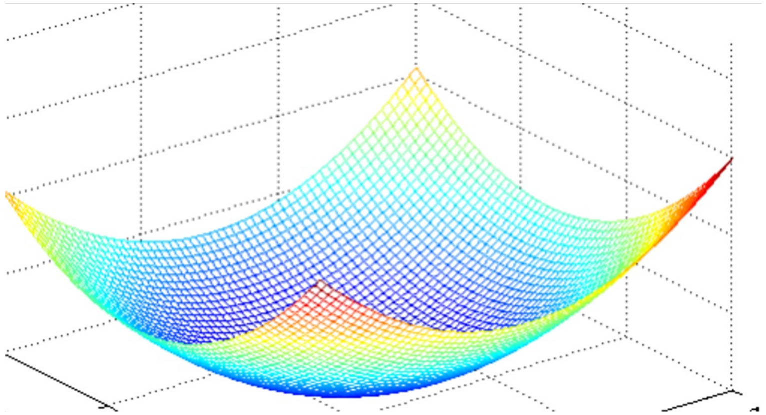
Strategy 1: Closed-Form Solution

- Recall that linear regression minimizes the loss

$$L(\beta; Z) = \frac{1}{n}\|Y - X\beta\|_2^2$$

- Minimum solution has gradient equal to zero:

$$\nabla_{\beta} L\left(\hat{\beta}(Z); Z\right) = 0$$



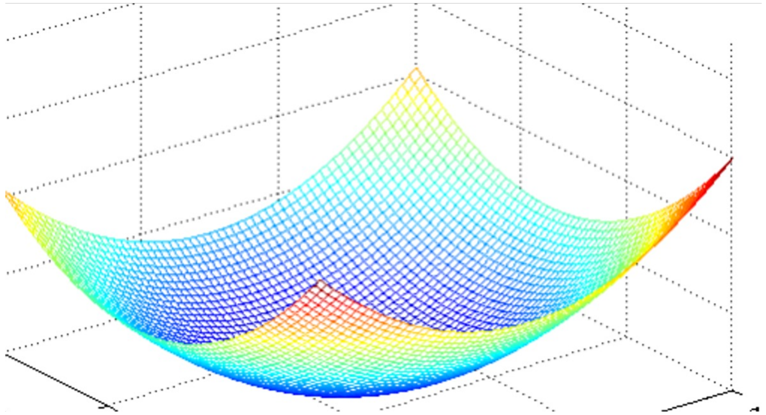
Strategy 1: Closed-Form Solution

- Recall that linear regression minimizes the loss

$$L(\beta; Z) = \frac{1}{n}\|Y - X\beta\|_2^2$$

- Minimum solution has gradient equal to zero:

$$\nabla_\beta L(\hat{\beta}; Z) = 0$$



Strategy 1: Closed-Form Solution

- The gradient is

$$\nabla_{\beta} L(\beta; Z)$$

Strategy 1: Closed-Form Solution

- The gradient is

$$\nabla_\beta L(\beta; Z) = \nabla_\beta \frac{1}{n} \|Y - X\beta\|_2^2$$

Strategy 1: Closed-Form Solution

- The gradient is

$$
\begin{aligned}
\nabla_\beta L(\beta; Z) = \nabla_\beta \frac{1}{n} \|Y - X\beta\|_2^2 &= \nabla_\beta \frac{1}{n} (Y - X\beta)^\top (Y - X\beta) \\
&= \frac{2}{n} \left[\nabla_\beta (Y - X\beta)^\top \right] (Y - X\beta) \\
&= -\frac{2}{n} X^\top (Y - X\beta) \\
&= -\frac{2}{n} X^\top Y + \frac{2}{n} X^\top X\beta
\end{aligned}
$$

Intuition on the Gradient

- By linearity of the gradient, we have

$$\nabla_\beta L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\beta (y_i - \beta^\top x_i)^2 = \frac{1}{n}\sum_{i=1}^{n} 2(y_i - \beta^\top x_i)x_i$$

- The gradient for a single term is

$$\nabla_\beta (y_i - \beta^\top x_i)^2 = 2(y_i - \beta^\top x_i)x_i$$

- I.e., the current error $y_i - \beta^\top x_i$ times the feature x_i

Strategy 1: Closed-Form Solution

- The gradient is

$$\nabla_\beta L(\beta; Z) = \nabla_\beta \frac{1}{n} \|Y - X\beta\|_2^2 = -\frac{2}{n} X^\top Y + \frac{2}{n} X^\top X\beta$$

- Setting $\nabla_\beta L(\hat{\beta}; Z) = 0$, we have $X^\top X\hat{\beta} = X^\top Y$

Strategy 1: Closed-Form Solution

- Setting $\nabla_\beta L(\hat{\beta}; Z) = 0$, we have $X^\top X \hat{\beta} = X^\top Y$
- Assuming $X^\top X$ is invertible, we have

$$\hat{\beta}(Z) = (X^\top X)^{-1} X^\top Y$$

Note on Invertibility

- Closed-form solution only **unique** if $X^\top X$ is invertible
  - Otherwise, **multiple solutions exist** to $X^\top X\hat{\beta} = X^\top Y$
  - **Intuition:** Underconstrained system of linear equations

- **Example:**

$$\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

  - In this case, any $\hat{\beta}_2 = 1 - \hat{\beta}_1$ is a solution

When Can this Happen?

- **Case 1**
  - Fewer data examples than feature dimension (i.e., $n < d$)
  - **Solution:** Remove features so $d \leq n$
  - **Solution:** Collect more data until $d \leq n$
  - **Solution:** Use L_1 regularization
- **Case 2:** Some feature is a linear combination of the others
  - **Special case (duplicated feature):** For some j and j', $x_{i,j} = x_{i,j'}$ for all i
  - **Solution:** Remove linearly dependent features
  - **Solution:** Use L_2 regularization

Shortcomings of Closed-Form Solution

- Computing $\hat{\beta}(Z) = (X^\top X)^{-1} X^\top Y$ can be challenging when the number of features d is large
- **Computing $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$ is $\boldsymbol{O}(\boldsymbol{d}^3)$**
  - $d = 10^4$ features → $O(10^{12})$
  - Even storing $X^\top X$ requires a lot of memory
- **Numerical accuracy issues due to "ill-conditioning"**
  - What if $X^\top X$ is "barely" invertible?
  - Then, $(X^\top X)^{-1}$ has large variance along some dimension
  - Regularization helps

Optimization Algorithms

- Recall that linear regression minimizes the loss

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2$$

- Iteratively optimize β
  - Initialize $\beta_1 \leftarrow \mathrm{Init}(\dots)$
  - For some number of iterations T, update $\beta_t \leftarrow \mathrm{Step}(\dots)$
  - Return β_T

Optimization Algorithms

- **Global search**: Try random values of β and choose the best
  - I.e., β_t independent of β_{t-1}
  - Very unstructured, can take a long time (especially in high dimension d)!
- **Local search**: Start from some initial β and make local changes
  - I.e., β_t is computed based on β_{t-1}
  - What is a "local change", and how do we find good one?

Strategy 2: Gradient Descent

- **Gradient descent:** Update β based on **gradient** $\nabla_\beta L(\beta; Z)$ of $L(\beta; Z)$:

$$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t; Z)$$

- **Intuition:** The gradient is the direction along which $L(\beta; Z)$ changes most quickly as a function of β
- $\alpha \in \mathbb{R}$ is a hyperparameter called the **learning rate**
  - More on this later

Strategy 2: Gradient Descent

- Choose initial value for β
- Until we reach a minimum:
  - Choose a new value for β to reduce $L(\beta; Z)$

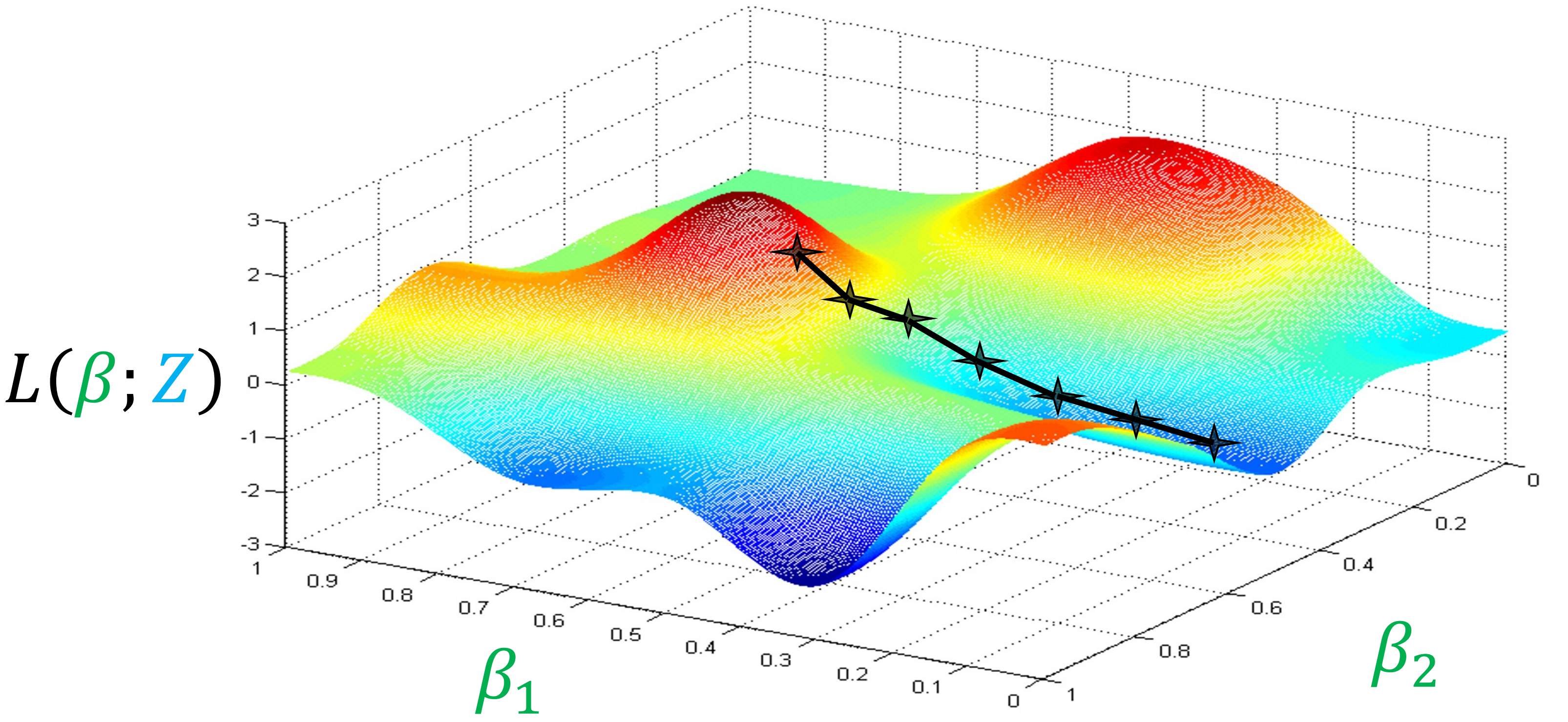


Figure by Andrew Ng

Strategy 2: Gradient Descent

- Choose initial value for β
- Until we reach a minimum:
  - Choose a new value for β to reduce $L(\beta; Z)$

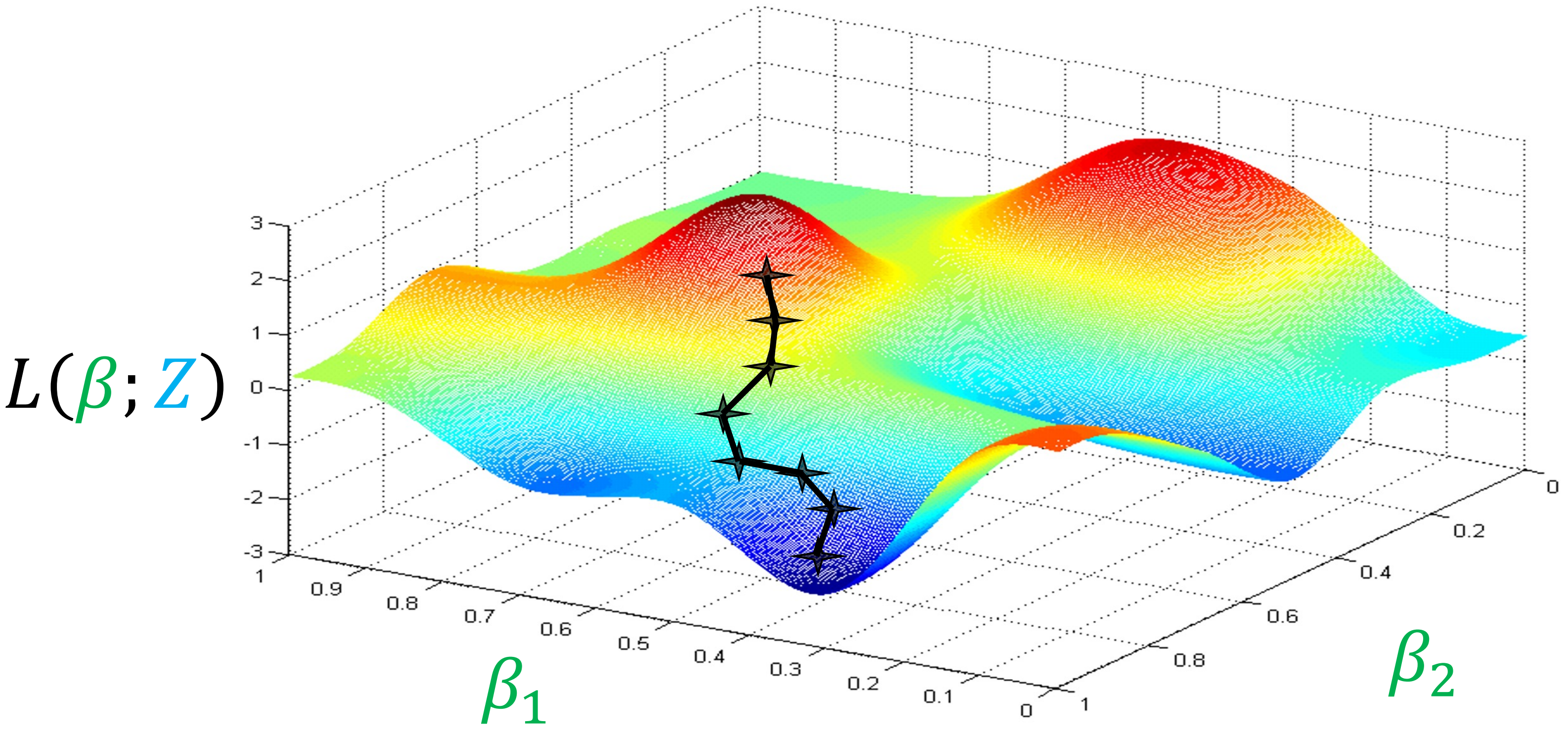
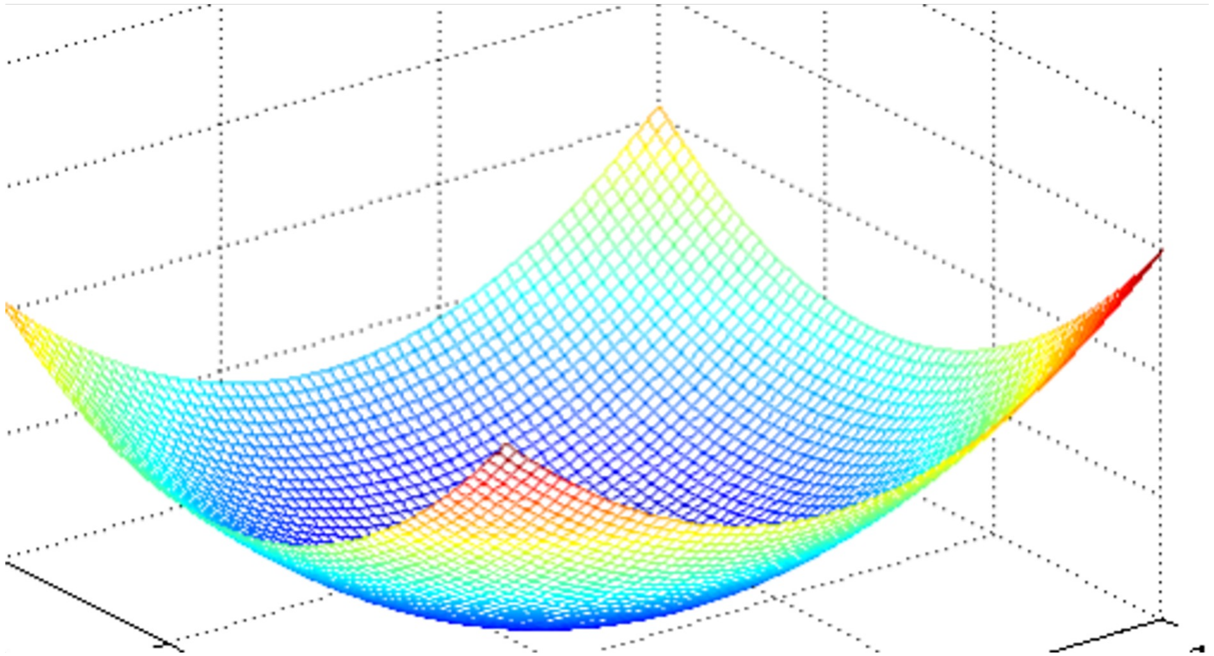


Figure by Andrew Ng

Strategy 2: Gradient Descent

- Choose initial value for β
- Until we reach a minimum:
  - Choose a new value for β to reduce $L(\beta; Z)$



Linear regression loss is convex, so no local minima

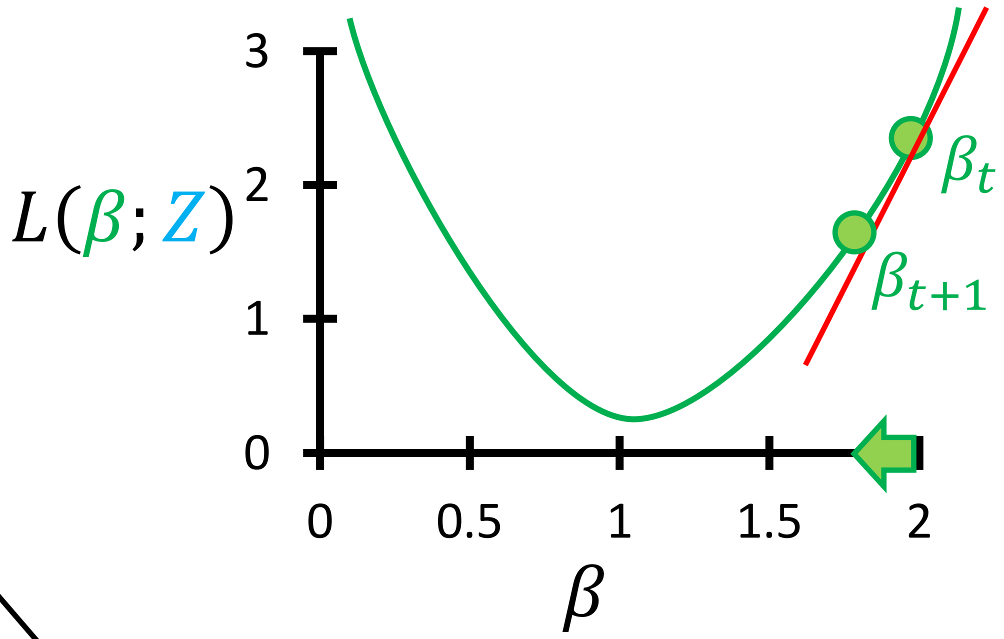
Figure by Andrew Ng

Strategy 2: Gradient Descent

- Initialize $\beta_1 = \vec{0}$
- Repeat until convergence:

$$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t; Z)$$

- For linear regression, know the gradient from strategy 1



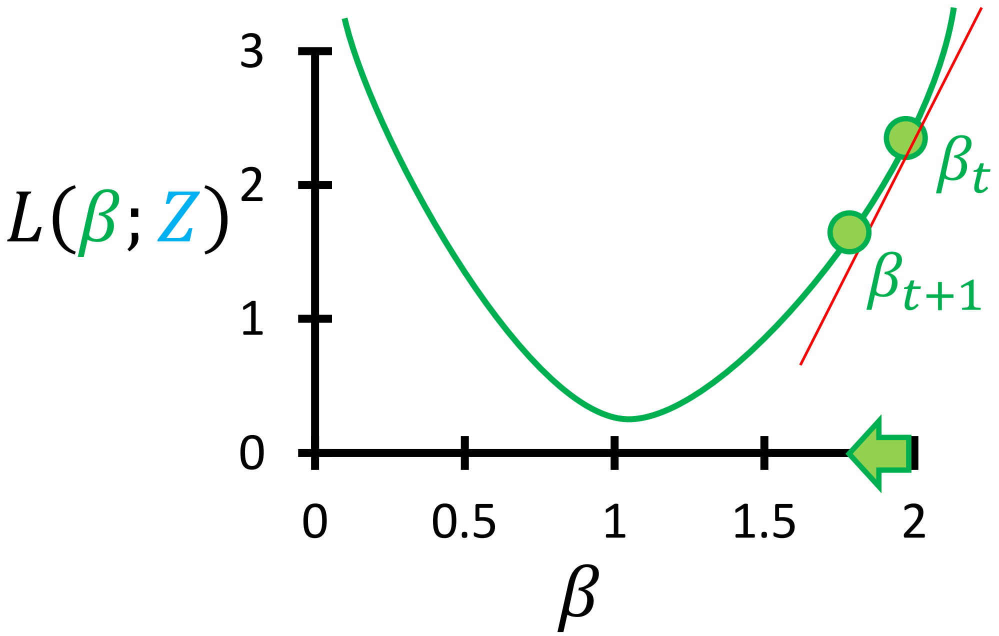
For in-place updates $\beta \leftarrow \beta - \alpha \cdot \nabla_\beta L(\beta; Z)$, compute all components of $\nabla_\beta L(\beta; Z)$ before modifying β

Strategy 2: Gradient Descent

- Initialize $\beta_1 = \vec{0}$
- Repeat until convergence:

$$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t; Z)$$

- For linear regression, know the gradient from strategy 1



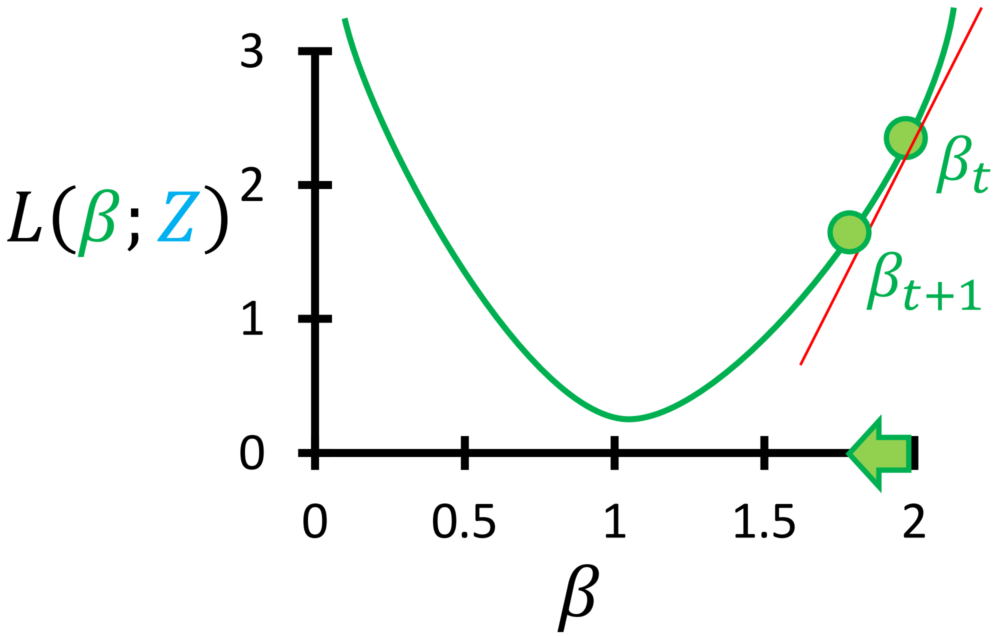
Strategy 2: Gradient Descent

- Initialize $\beta_1 = \vec{0}$
- Repeat until $\|\beta_t - \beta_{t+1}\|_2 \leq \epsilon$:

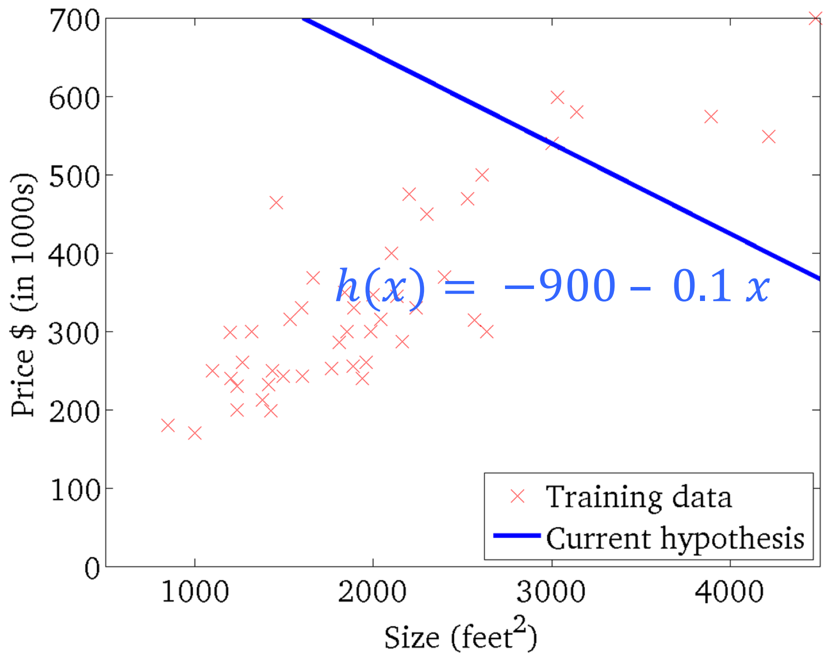
Hyperparameter defining convergence

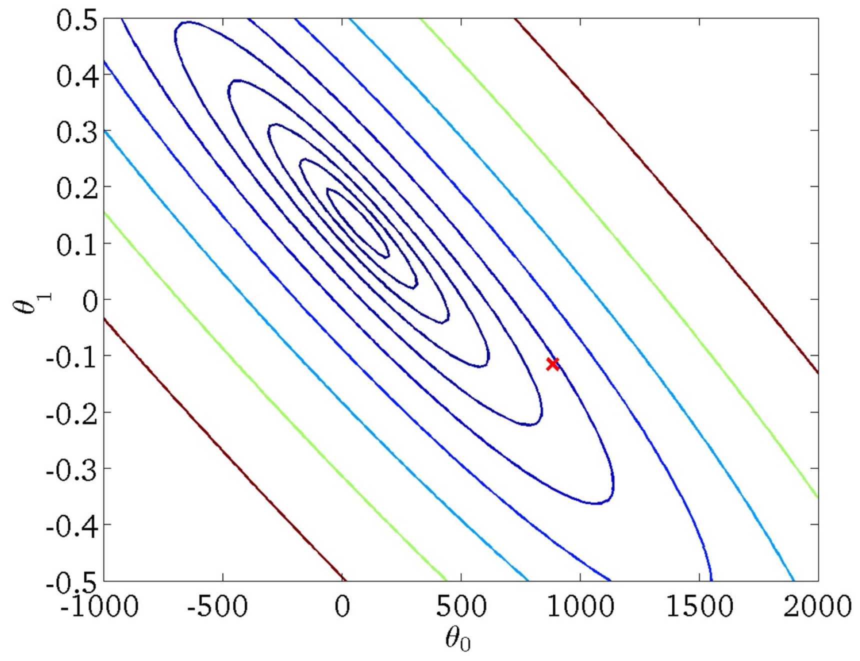
$$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t; Z)$$

- For linear regression, know the gradient from strategy 1



Strategy 2: Gradient Descent

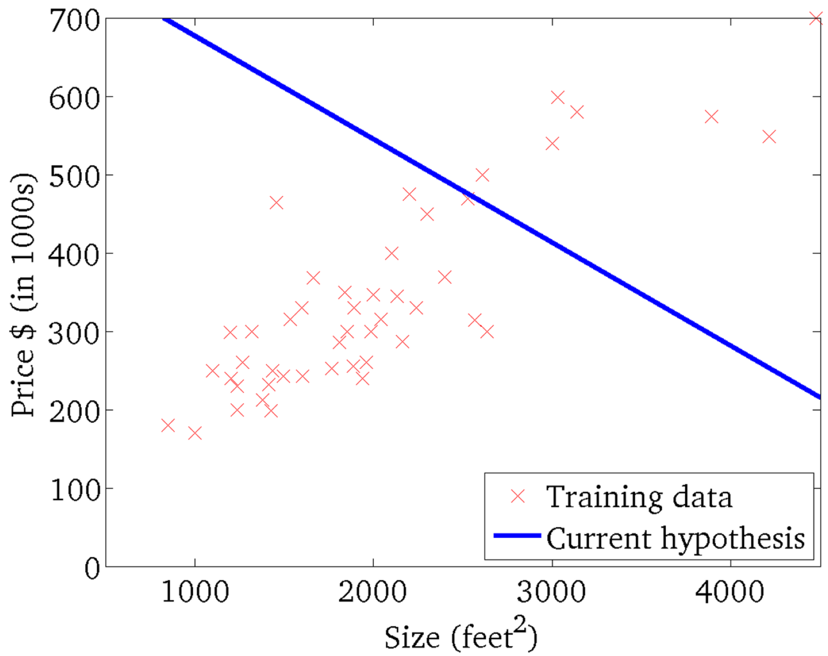


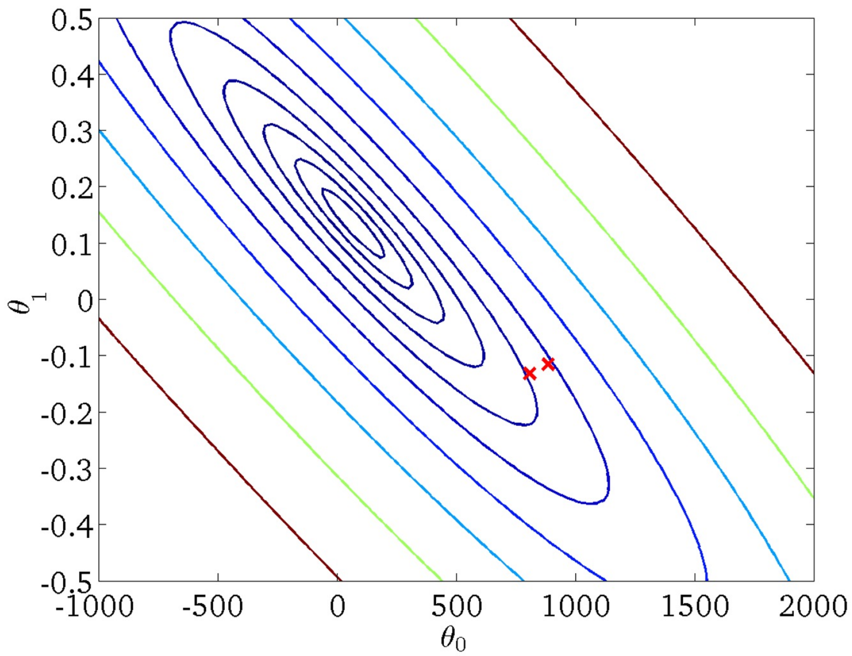


$f_\beta(x)$ $L(\beta; Z)$

Slide by Andrew Ng

Strategy 2: Gradient Descent

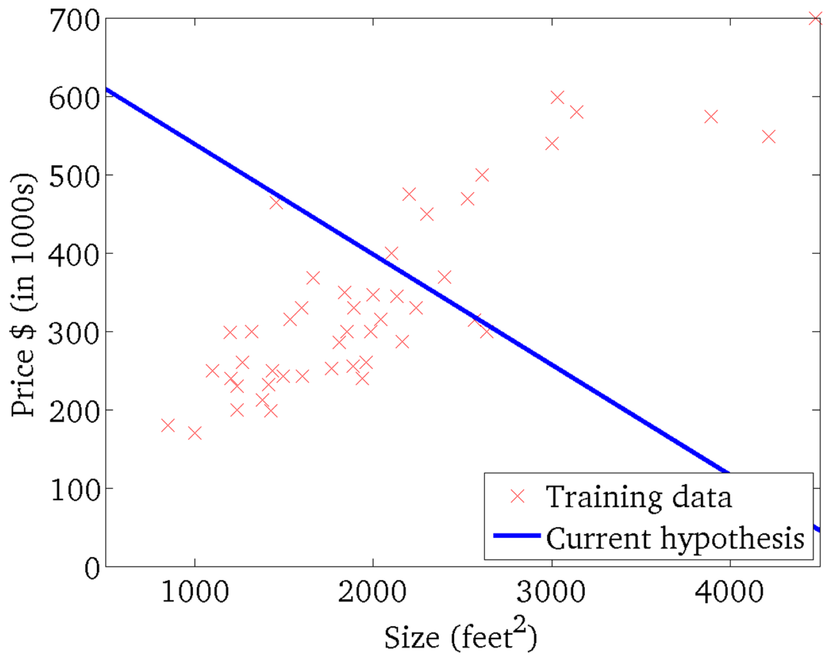




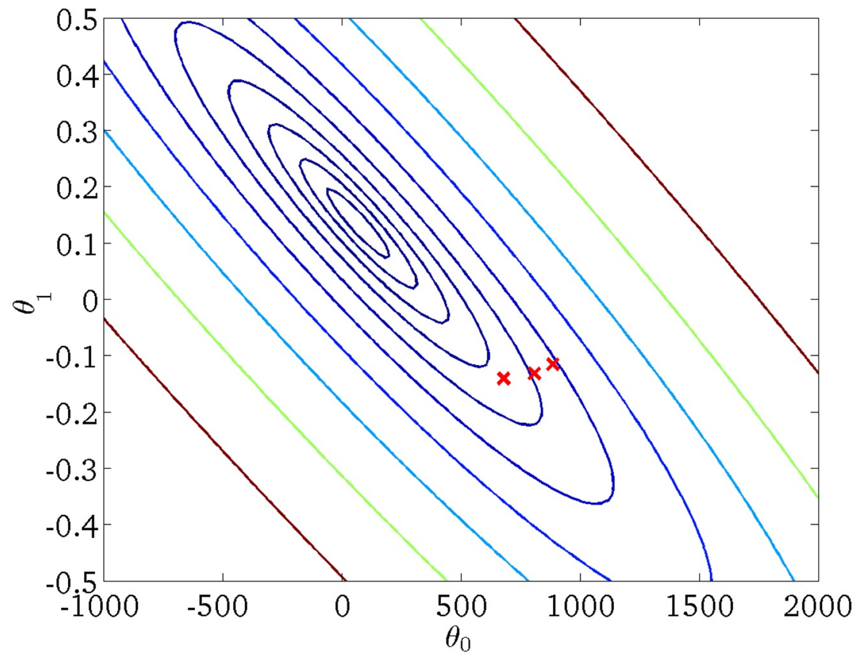
$f_\beta(x)$ $L(\beta; Z)$

Slide by Andrew Ng

Strategy 2: Gradient Descent



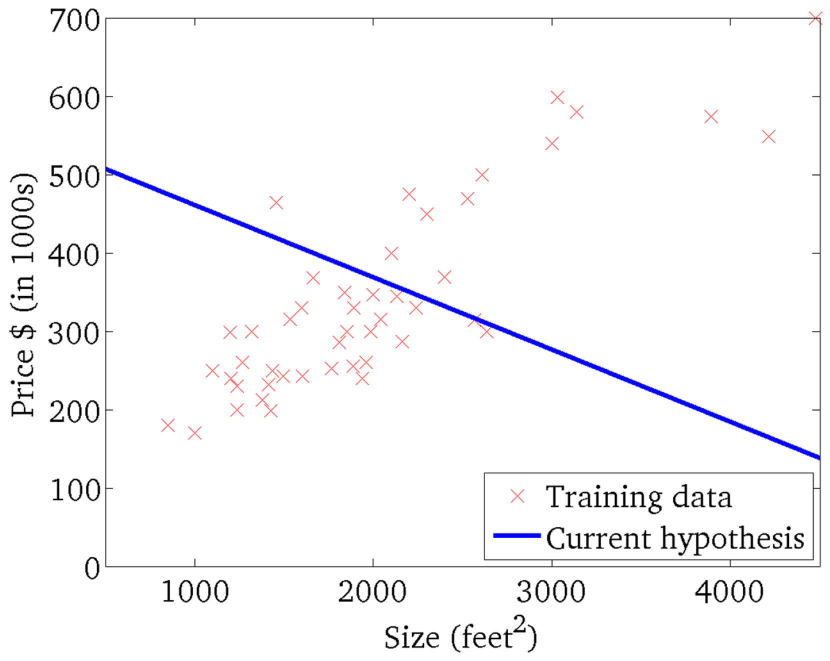
$f_\beta(x)$



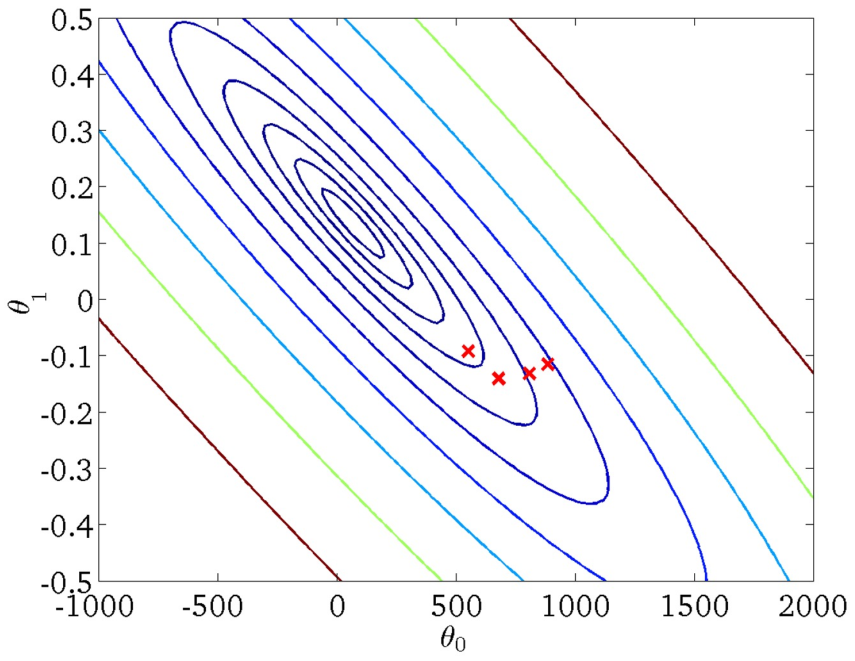
$L(\beta; Z)$

Slide by Andrew Ng

Strategy 2: Gradient Descent

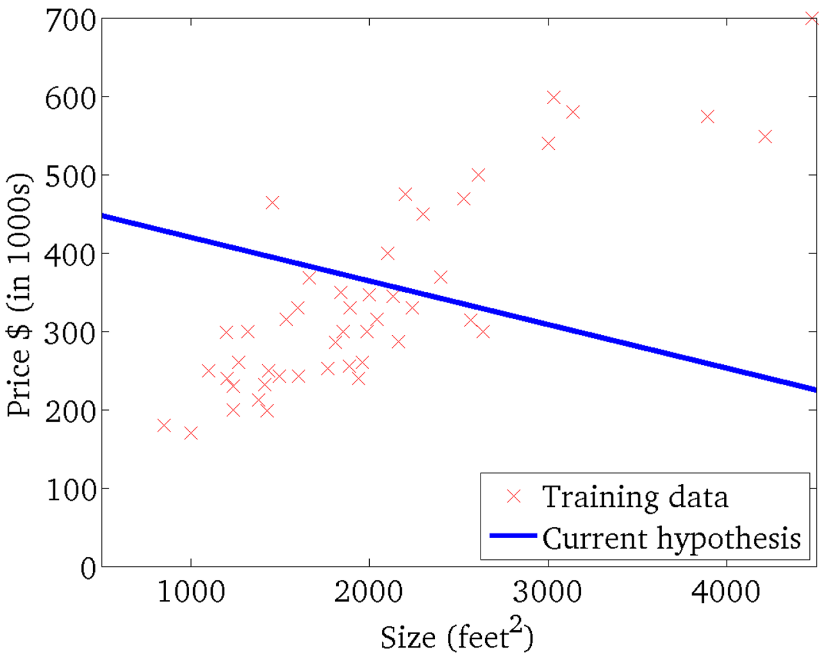


$f_\beta(x)$

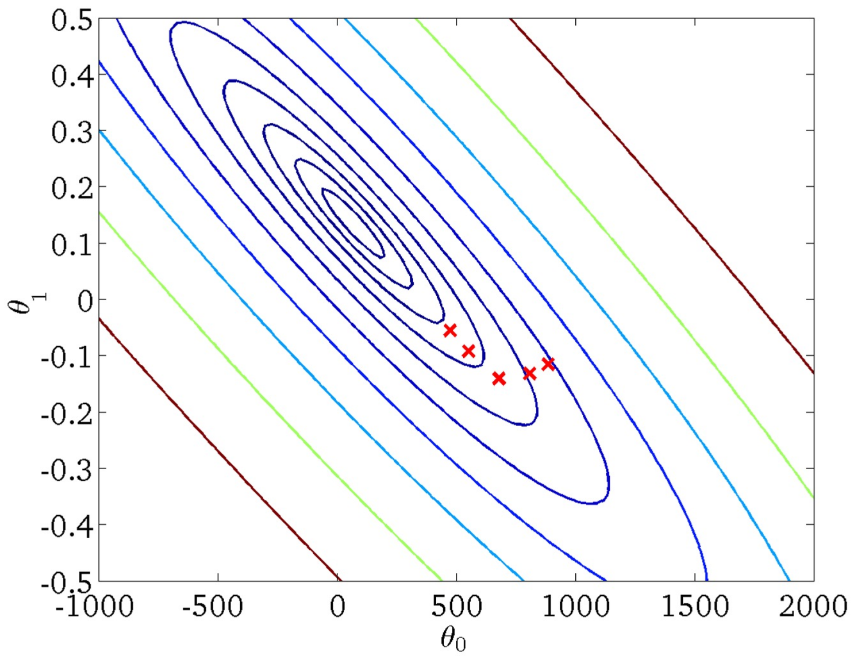


$L(\beta; Z)$

Strategy 2: Gradient Descent



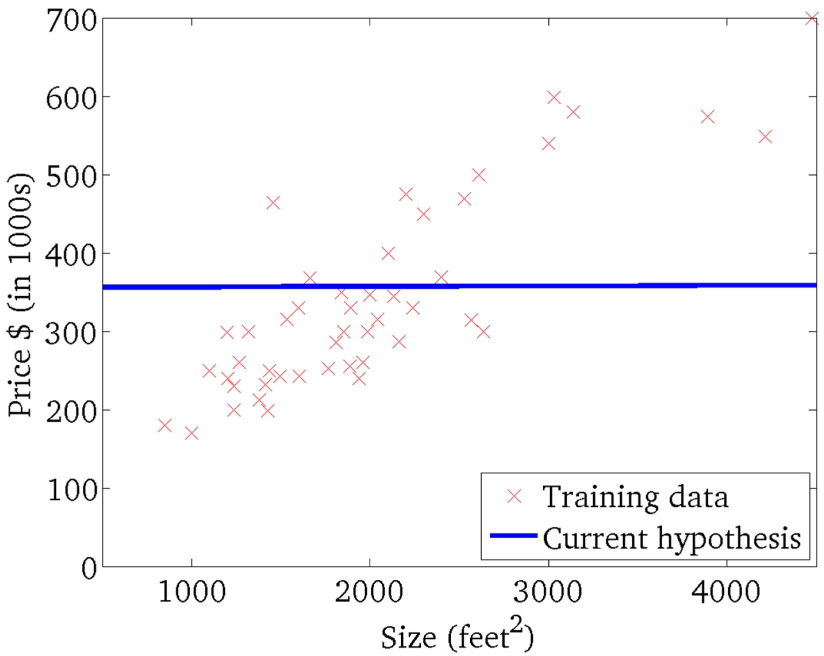
$f_\beta(x)$

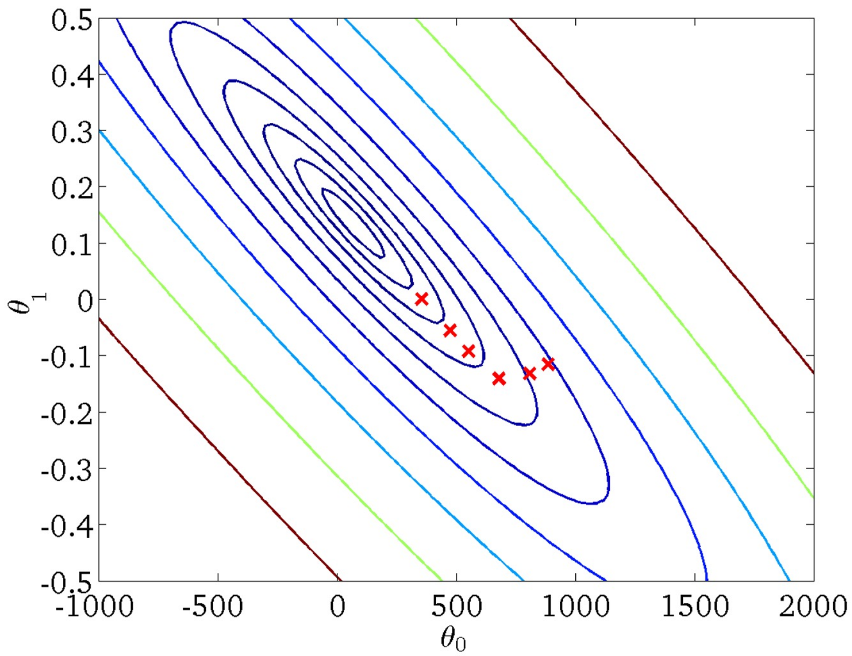


$L(\beta; Z)$

Slide by Andrew Ng

Strategy 2: Gradient Descent

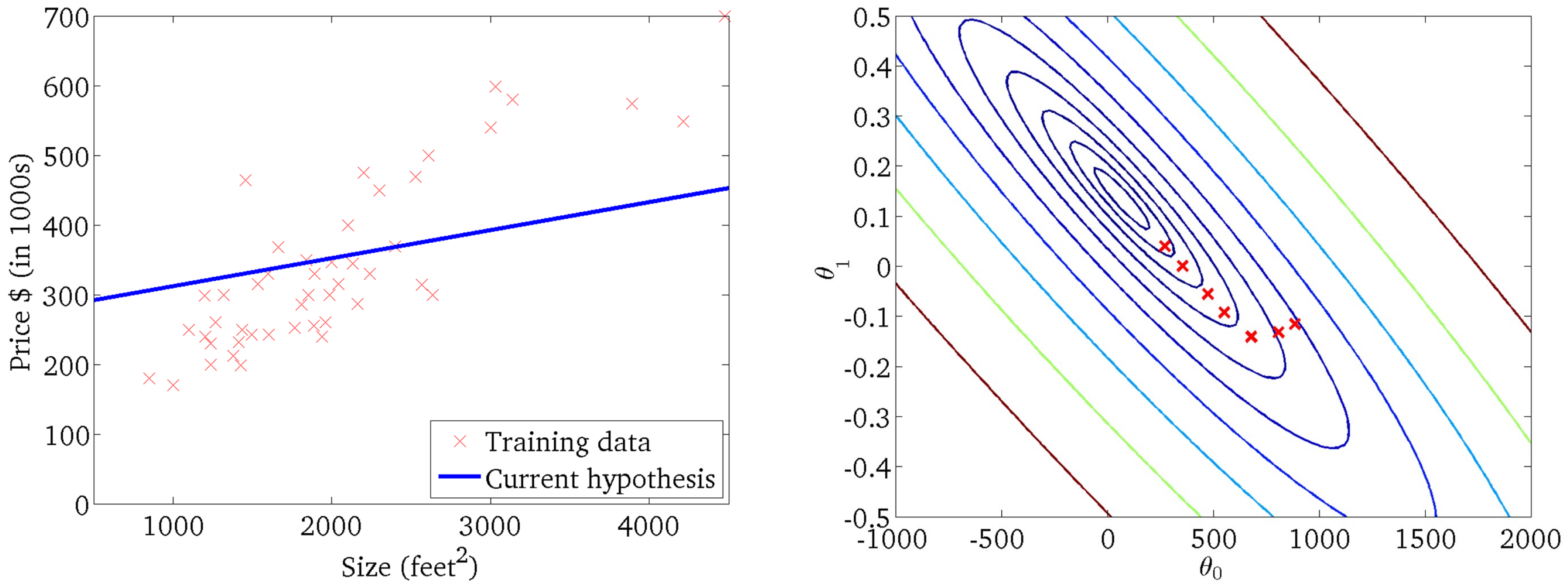




$f_\beta(x)$ $\qquad$ $L(\beta; Z)$

Slide by Andrew Ng

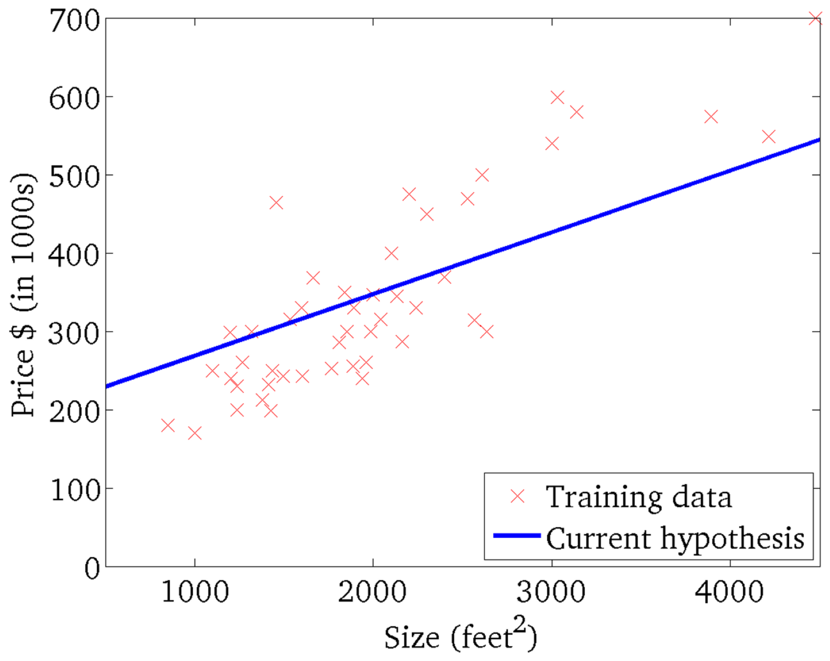
Strategy 2: Gradient Descent



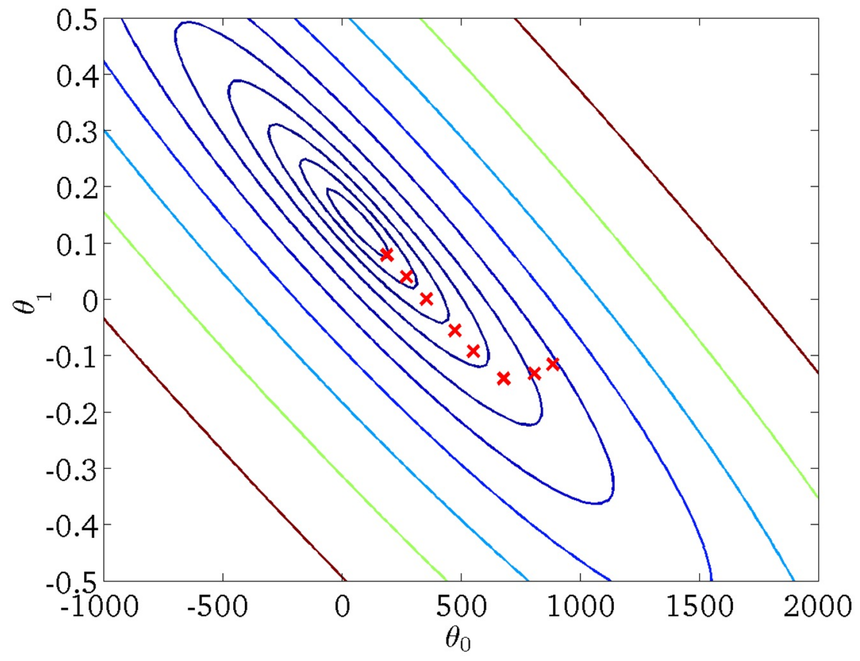
$f_\beta(x)$

$L(\beta; Z)$

Strategy 2: Gradient Descent



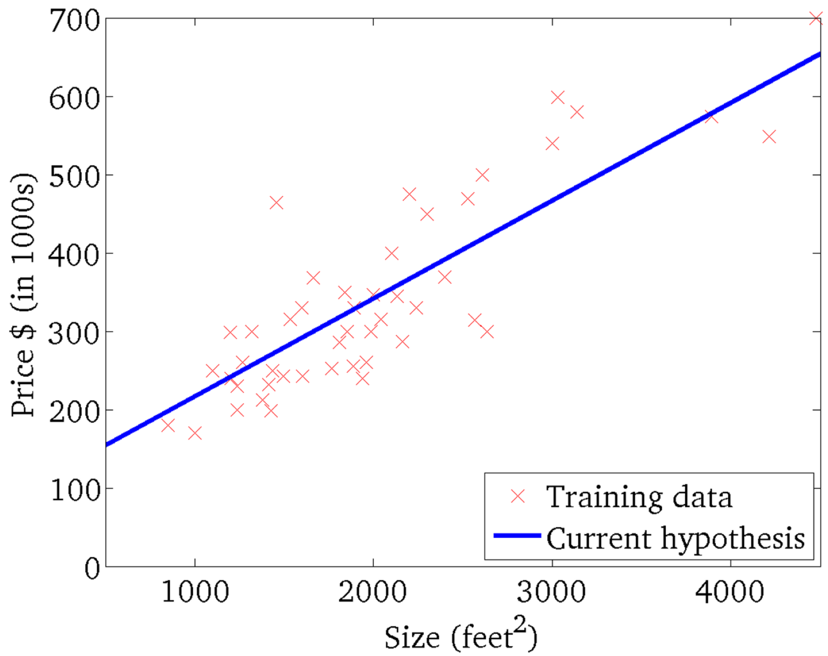
$$f_\beta(x)$$



$$L(\beta; Z)$$

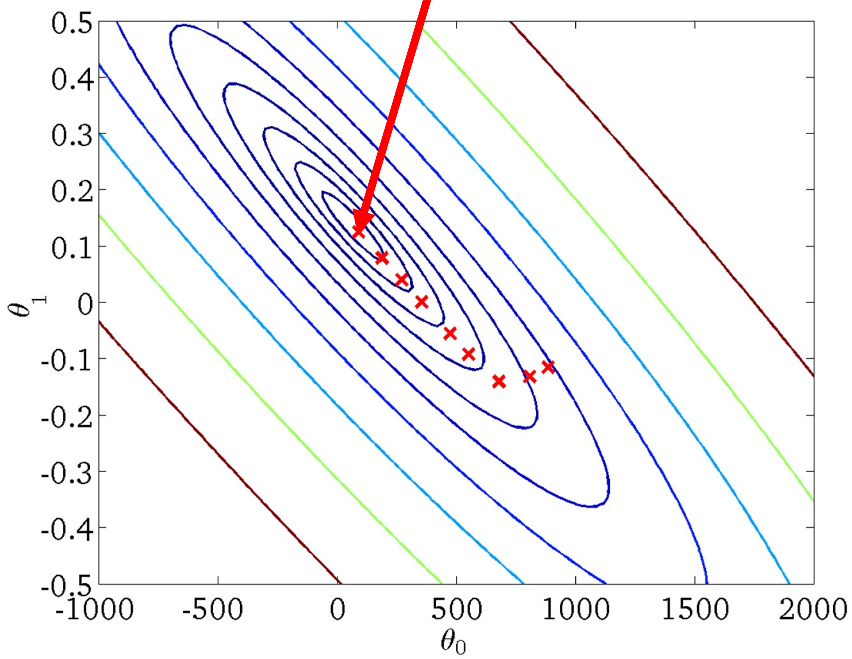
Slide by Andrew Ng

Strategy 2: Gradient Descent



$f_\beta(x)$

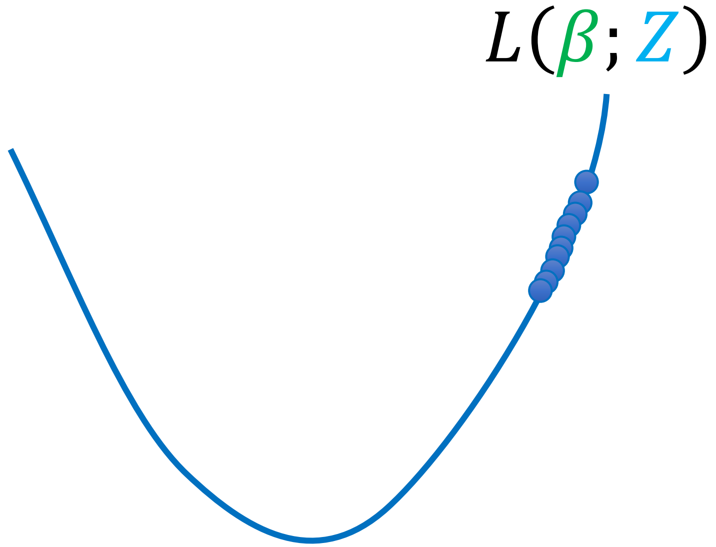
Minimizer of loss function



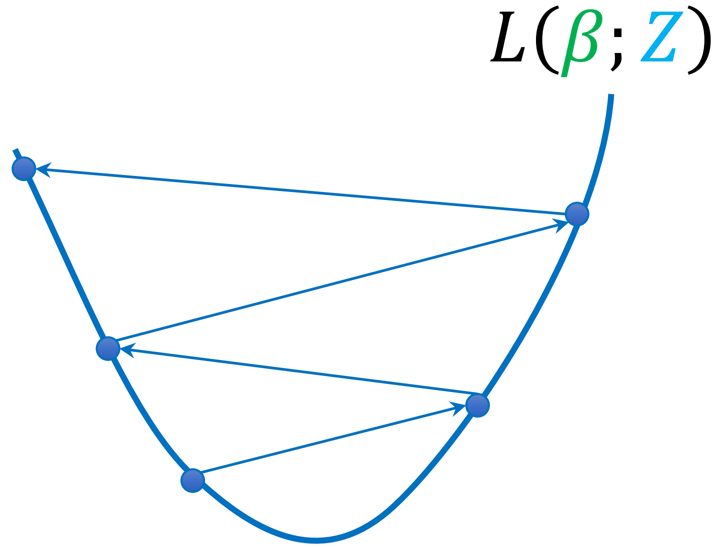
$L(\beta; Z)$

Slide by Andrew Ng

Choice of Learning Rate $\boldsymbol{\alpha}$



Problem: α too small
- $L(\beta; Z)$ decreases slowly



Problem: α too large
- $L(\beta; Z)$ increases!

Plot $L(\beta_t; Z_{\text{train}})$ vs. t to diagnose these problems

Choice of Learning Rate $\boldsymbol{\alpha}$

- α is a hyperparameter for gradient descent that we need to choose
  - Can set just based on training data

- **Rule of thumb**
  - **$\boldsymbol{\alpha}$ too small:** Loss decreases slowly
  - **$\boldsymbol{\alpha}$ too large:** Loss increases!

- Try rates $\alpha \in \{1.0, 0.1, 0.01, \dots\}$ (can tune further once one works)

Comparison of Strategies

- **Closed-form solution**
  - No hyperparameters
  - Slow if n or d are large
- **Gradient descent**
  - Need to tune α
  - Scales to large n and d
- For linear regression, there are better optimization algorithms, but gradient descent is very general
  - Accelerated gradient descent is an important tweak that improves performance in practice (and in theory)

$\boldsymbol{L_2}$ Regularized Linear Regression

- Recall that linear regression with L_2 regularization minimizes the loss

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \sum_{j=1}^{d} \beta_j^2$$

L_2 Regularized Linear Regression

- Recall that linear regression with L_2 regularization minimizes the loss

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \sum_{j=1}^{d} \beta_j^2 = \frac{1}{n}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$$

- Gradient is

$$\nabla_\beta L(\beta; Z) = -\frac{2}{n}X^\top Y + \frac{2}{n}X^\top X\beta + 2\lambda\beta$$

Strategy 1: Closed-Form Solution

- Gradient is

$$\nabla_\beta L(\beta; Z) = -\frac{2}{n} X^\top Y + \frac{2}{n} X^\top X \beta + 2\lambda\beta$$

- Setting $\nabla_\beta L(\hat{\beta}; Z) = 0$, we have $(X^\top X + n\lambda I)\hat{\beta} = X^\top Y$
- Always invertible if $\lambda > 0$, so we have

$$\hat{\beta}(Z) = (X^\top X + n\lambda I)^{-1} X^\top Y$$

Strategy 2: Gradient Descent

- Gradient is

$$\nabla_\beta L(\beta; Z) = -\frac{2}{n} X^\top Y + \frac{2}{n} X^\top X\beta + 2\lambda\beta$$

- Same algorithm as vanilla linear regression (a.k.a. OLS)
- **Intuition:** The extra term $\lambda\beta$ in the gradient is **weight decay** that encourages β to be small

What About $\boldsymbol{L_1}$ Regularization?

- Gradient descent still works!

- Specialized algorithms work better in practice
  - **Simple one:** Gradient descent + soft thresholding
  - Basically, if $|\beta_{t,j}| \leq \lambda$, just set it to zero
  - Good theoretical properties

Loss Minimization View of ML

- **Two design decisions**
  - **Model family:** What are the candidate models f? (E.g., linear functions)
  - **Loss function:** How to define "approximating"? (E.g., MSE loss)

Loss Minimization View of ML

- **Three design decisions**
  - **Model family:** What are the candidate models f? (E.g., linear functions)
  - **Loss function:** How to define “approximating”? (E.g., MSE loss)
  - **Optimizer:** How do we minimize the loss? (E.g., gradient descent)

Lecture 5: Logistic Regression

CIS 4190/5190

Fall 2022

Supervised Learning

Data $Z = \{(x_i, y_i)\}_{i=1}^{n}$

$\hat{\beta}(Z) = \arg\min_{\beta} L(\beta; Z)$

L encodes $y_i \approx f_{\beta}(x_i)$

Model $f_{\hat{\beta}(Z)}$

Regression

Data $Z = \{(x_i, y_i)\}_{i=1}^{n}$

$\hat{\beta}(Z) = \arg\min_{\beta} L(\beta; Z)$

L encodes $y_i \approx f_{\beta}(x_i)$

Model $f_{\hat{\beta}(Z)}$

Label is a **real value** $y_i \in \mathbb{R}$

Classification

Data $Z = \{(x_i, y_i)\}_{i=1}^{n}$

$\hat{\beta}(Z) = \arg\min_\beta L(\beta; Z)$

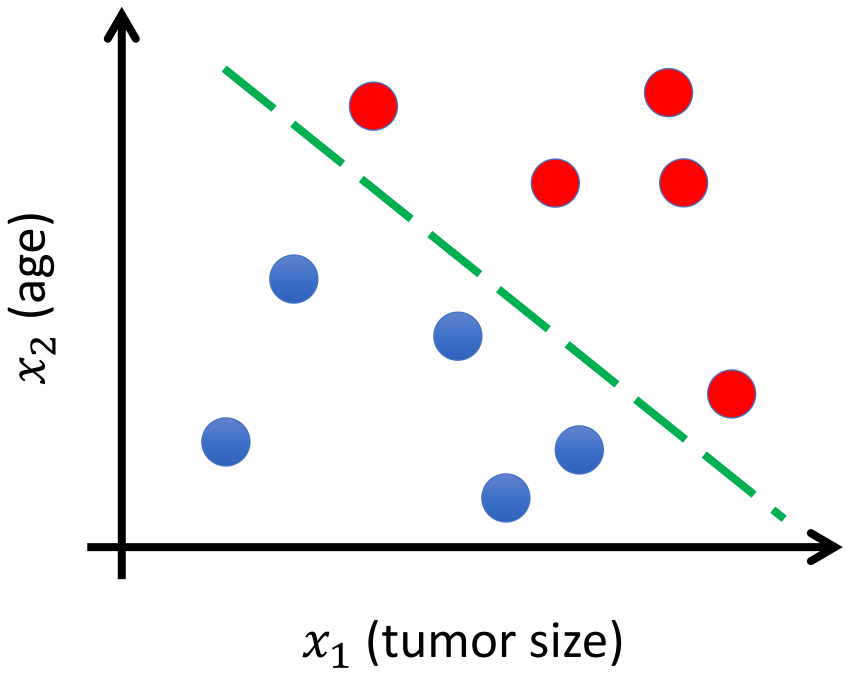
L encodes $y_i \approx f_\beta(x_i)$

Model $f_{\hat{\beta}(Z)}$

Label is a **discrete value** $y_i \in \mathcal{Y} = \{c_1, \ldots, c_k\}$

(Binary) Classification

- **Input:** Dataset $Z = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
- **Output:** Model $y_i \approx f_\beta(x_i)$



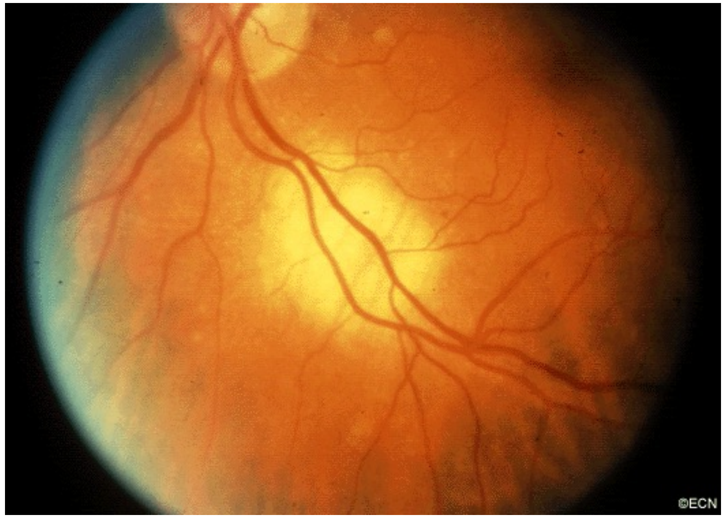


Image: https://eyecancer.com/uncategorized/choroidal-metastasis-test/

Example: Malignant vs. Benign Ocular Tumor

Loss Minimization View of ML

- **Three design decisions**
  - **Model family:** What are the candidate models f? (E.g., linear functions)
  - **Loss function:** How to define "approximating"? (E.g., MSE loss)
  - **Optimizer:** How do we optimize the loss? (E.g., gradient descent)

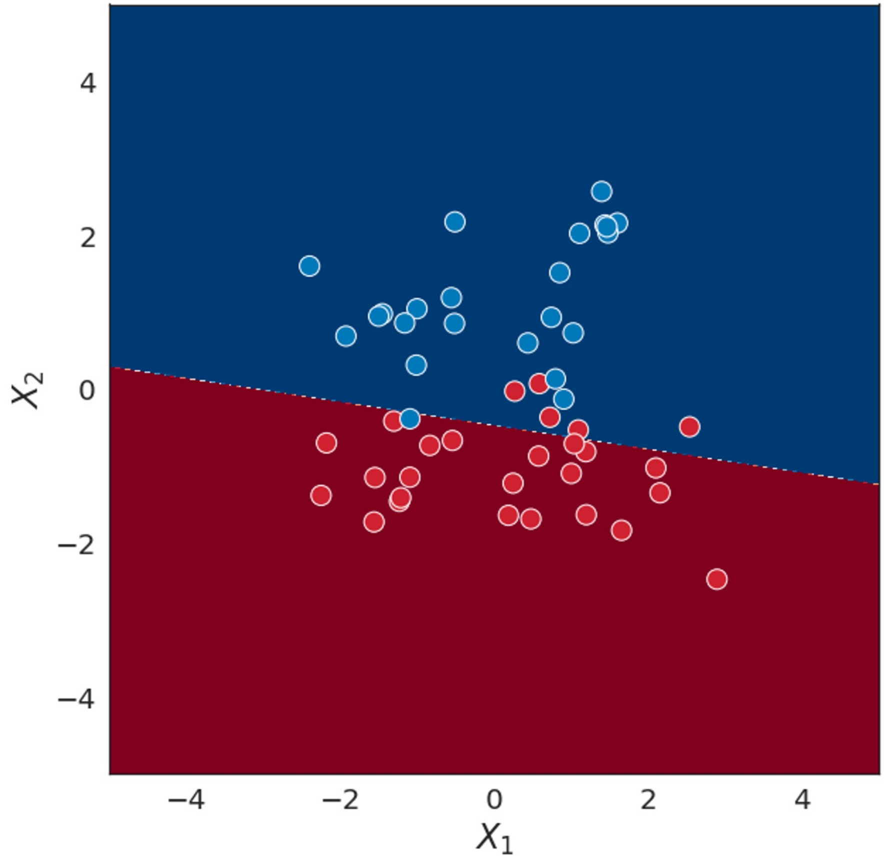
- How do we adapt to classification?

Linear Functions for (Binary) Classification

- **Input:** Dataset $Z = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$

- **Regression:**
  - Labels $y_i \in \mathbb{R}$
  - Predict $y_i \approx \beta^\top x_i$

- **Classification:**
  - Labels $y_i \in \{0, 1\}$
  - Predict $y_i \approx 1(\beta^\top x_i \geq 0)$
  - $1(C)$ equals 1 if C is true and 0 if C is false
  - How to learn β? **Need a loss function!**

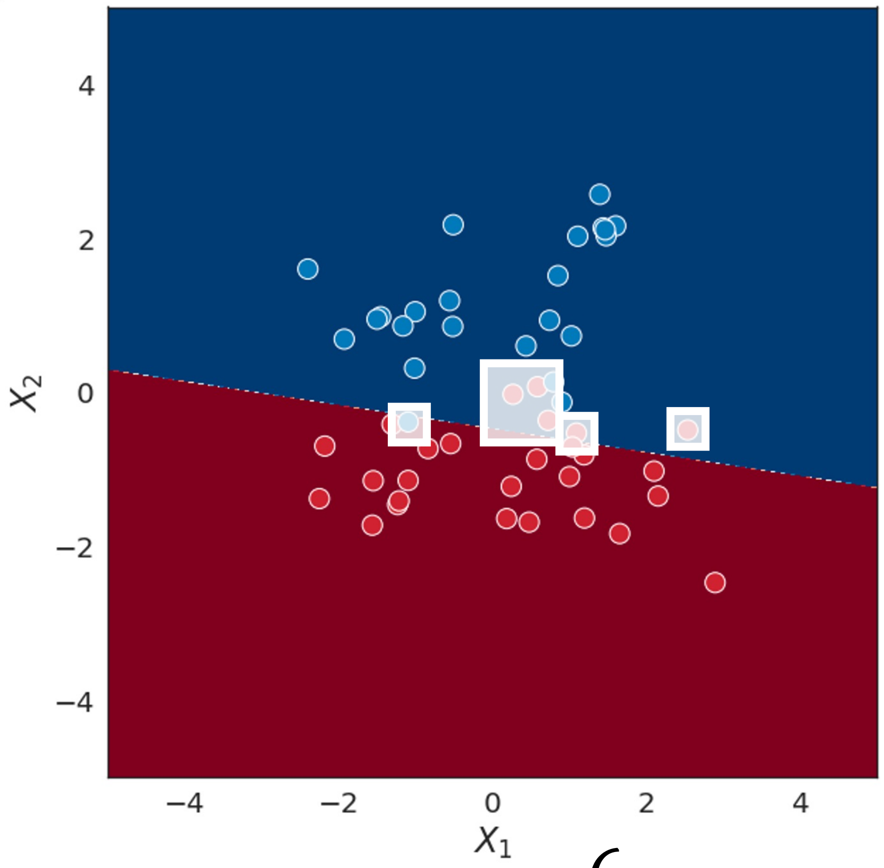


Loss Functions for Linear Classifiers

- **(In)accuracy:**

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n} 1\left(y_i \neq f_\beta(x_i)\right)$$

- Computationally intractable
- Often, but not always the "true" loss (e.g., imbalanced data)



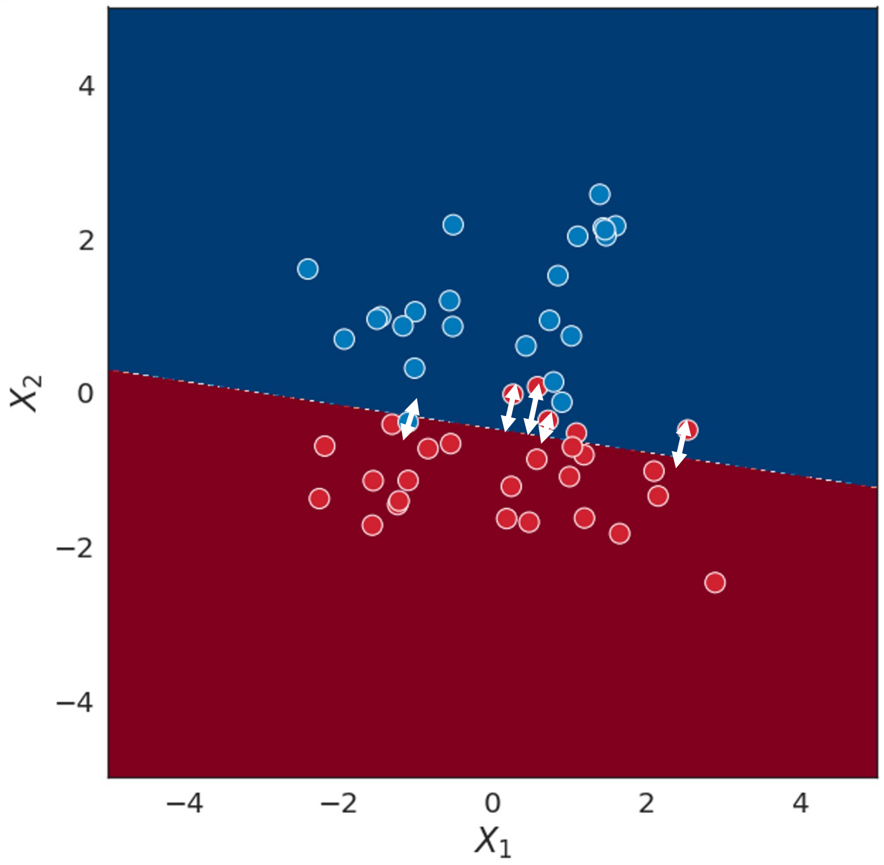
$$L(\beta; Z) = \frac{6}{50}$$

Loss Functions for Linear Classifiers

- **Distance:**

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{dist}(x_i, f_\beta) \cdot 1\big(f_\beta(x_i) \neq y_i\big)$$

- If $L(\beta; Z) = 0$, then 100% accuracy
- Variant of this loss results in SVM
- But, we will consider a more general strategy



$$L(\beta; Z) = 1.2$$