50.007 Machine Learning

Lecture 4

Regression

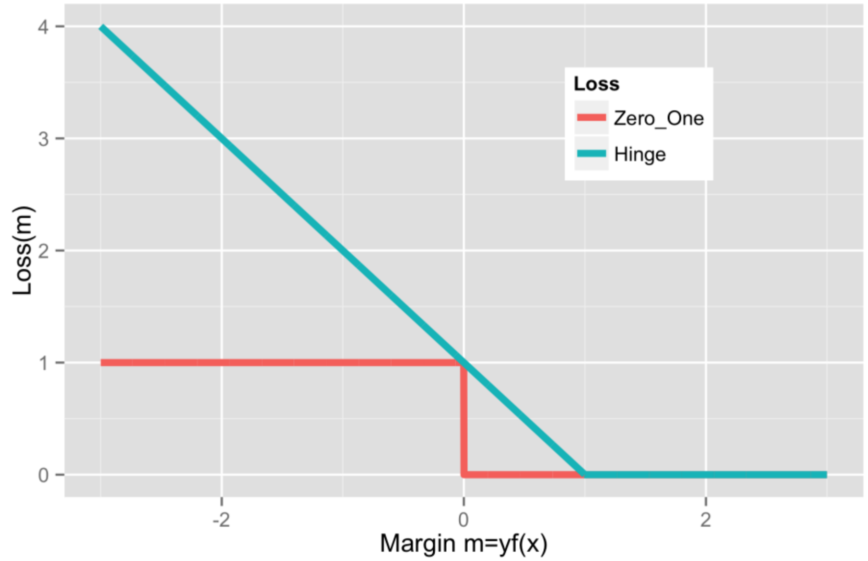
Recap

Loss Functions

- **Empirical risk:** $R_n(\theta) = \frac{1}{n} \sum_{\text{data } (x,y)} \text{Loss}\big(y(\theta^\top x)\big)$

- **Zero-one loss:** $\text{Loss}_{0|1}(z) = [\![\, z \le 0 \,]\!]$

- **Hinge loss:** $\text{Loss}_{\text{h}}(z) = \max\{1 - z, 0\}$



CONVEX!
Penalize larger mistakes more.
Penalize near-mistakes, i.e. $0 \le z \le 1$.

Stochastic Gradient Descent

1. Initialize the **weight** ($\theta^{(0)} = 0$).

2. Select $t \in \{1, \ldots, n\}$ at $\boxed{\text{random}}$

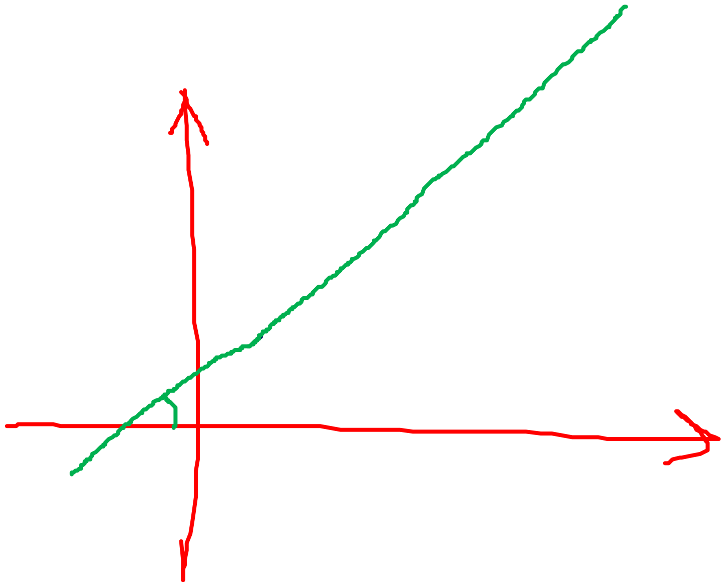
max$(1-z, 0)$

- If $\boxed{y^{(t)}(\theta^{(k)} \cdot x^{(t)}) \leq 1}$, then update the weight

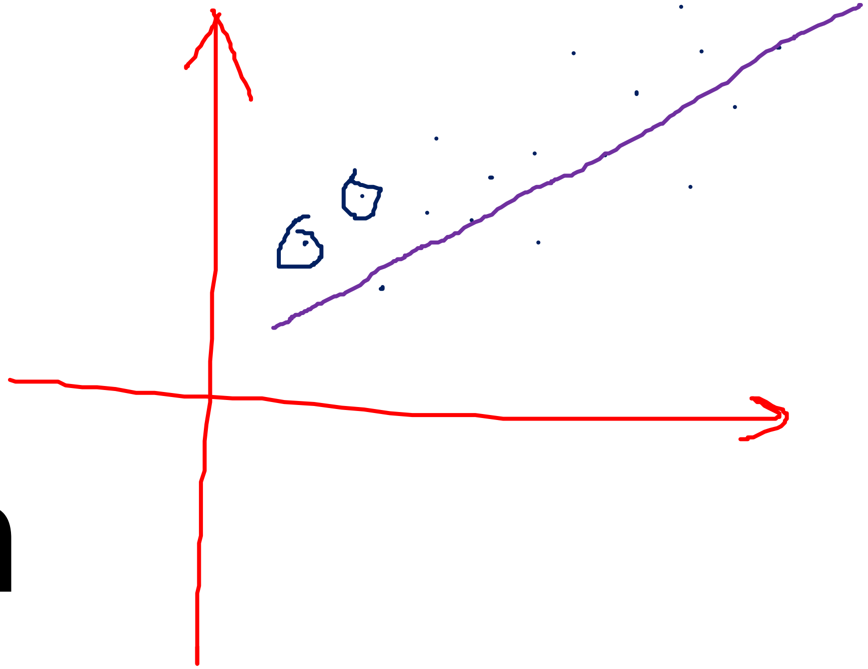
$$\boxed{\theta^{(k+1)} = \theta^{(k)} + \eta_k y^{(t)} x^{(t)}}$$

0

3. Repeat Step (2) until stopping criterion is met.
(e.g. when improvement in $R_n(\theta)$ is small enough)



Linear Regression



Machine Learning

Task

Performance

Experience

Algorithms that improve their **performance** at some **task** with **experience**
– Tom Mitchell (1998)

Linear Regression

Machine Learning

> Supervised Learning

> Classification

> Regression

- **Task.** Find function $f: \mathbb{R}^d \rightarrow \mathbb{R}$ such that $y \approx f(x; \theta)$
- **Experience.** Training data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$
- **Performance.** Prediction error $(y - f(x; \theta))$ on test data

Linear Regression

Training data

$$\mathcal{S}_n = \{ (x^{(t)}, y^{(t)}) \mid t = 1, \dots, n \}$$

- Features/Inputs $x^{(t)} = \left(x_1^{(t)}, \dots, x_d^{(t)}\right)^\mathsf{T} \in \mathbb{R}^d$
- Response/Output $y^{(t)} \in \mathbb{R}$

$$\begin{bmatrix} x_1^t \\ x_2^t \\ \vdots \\ x_d^t \end{bmatrix}$$

Linear Regression

Model (or Hypothesis Class) F

Each f is a *predictor* or *hypothesis*

Set of *linear* functions $f: \mathbb{R}^d \rightarrow \mathbb{R}$

$$f(x; \theta, \theta_0) = \boxed{\theta . x} = \theta_d x_d + \cdots + \theta_1 x_1 + \theta_0 = \theta^\top x + \theta_0$$

Model Parameters

$(1, x_1, x_2, \ldots, x_d)$

$$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

Least Square Loss

Loss Function $\text{Loss}(z) = \frac{1}{2}z^2 = \frac{1}{2}\left(y^{(t)} - \left(\theta . x^{(t)}\right)\right)^2$

Squared error.
Penalize big errors more heavily.
CONVEX!!

Empirical Risk

$$R_1(\theta; x, y) \quad = \text{Loss}\left(y^{(t)} - \left(\theta . x^{(t)}\right)\right)$$

Point loss

$$R_n(\theta; \mathcal{S}_n) \quad = \frac{1}{n} \sum_{(x,y)\in\mathcal{S}_n} R_1(\theta; x, y)$$

Average loss

$$= \frac{1}{n} \sum_{(x,y)\in\mathcal{S}_n} \frac{1}{2}\left(y^{(t)} - \left(\theta . x^{(t)}\right)\right)^2$$

The training loss is the average of the point losses.

Risk = “Expected Loss”
Empirical = “of the Data”

Linear Regression

- **Empirical Risk and Least Squares Criterion**

Training Loss/Objective

$$R_n(\theta) = \frac{1}{n}\sum_{t=1}^{n} \text{Loss}(y^{(t)} - \theta \cdot x^{(t)}) = \frac{1}{n}\sum_{t=1}^{n}(y^{(t)} - \theta \cdot x^{(t)})^2/2$$

Training Algorithm

Find predictor $f \in F$ that minimizes $R_n(\theta)$

The test loss and training loss can be different.

Linear Regression

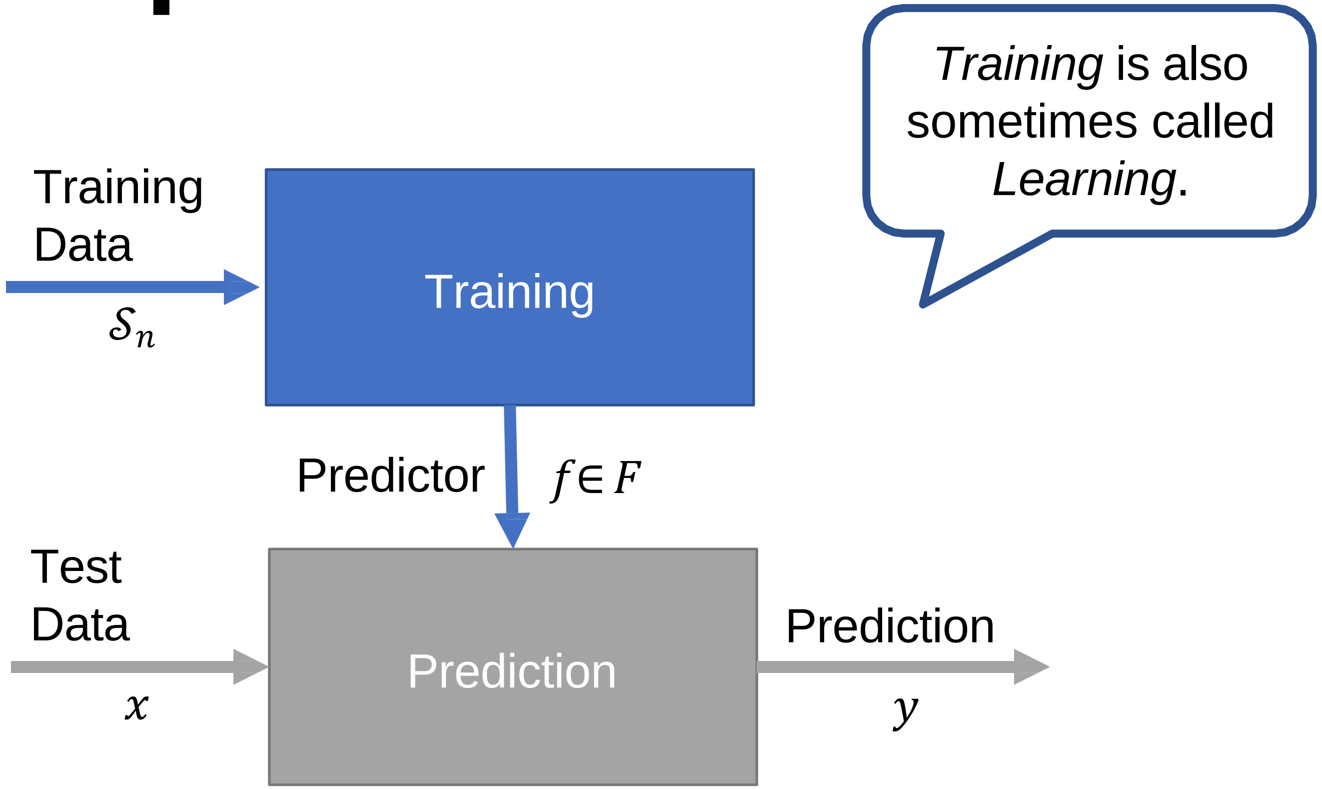
- **Empirical Risk and Least Squares Criterion**

Test Loss/Objective

$$R_{n'}^{test}(\theta) = \frac{1}{|Sn'|} \sum_{t=n+1}^{n+n'} (y^{(t)} - \theta \cdot x^{(t)})^2/2$$

Given a predictor f, we use the test loss to measure how well it generalizes to new data.

Training and prediction



Assumption. Test data and training data are identically distributed.

Generalization

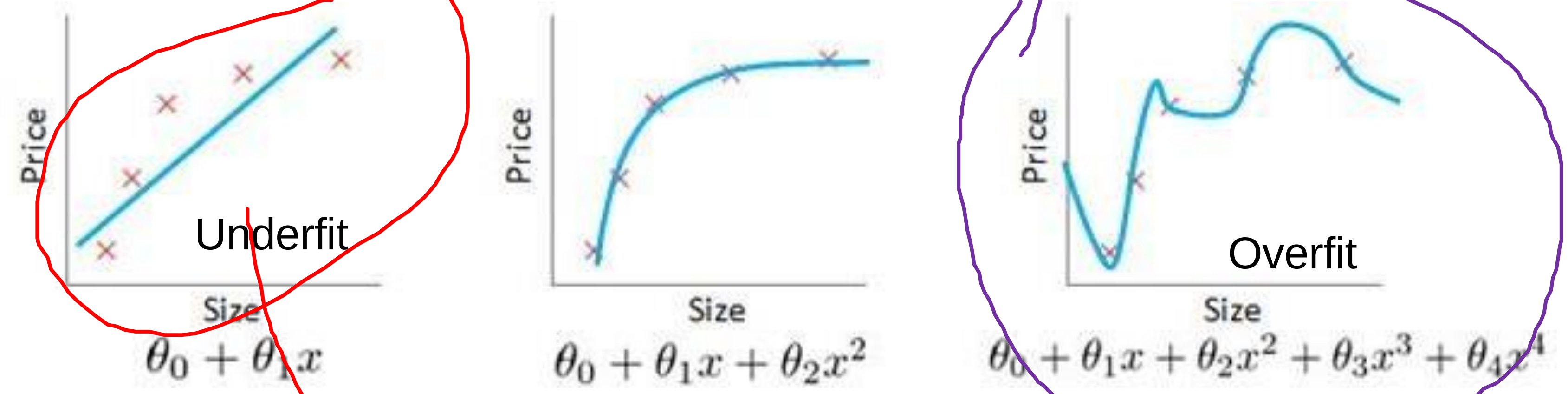
The goal of machine learning is to find a predictor $f \in F$ that generalizes well, i.e. that predicts well on test data $\mathcal{S}_{n'}$.

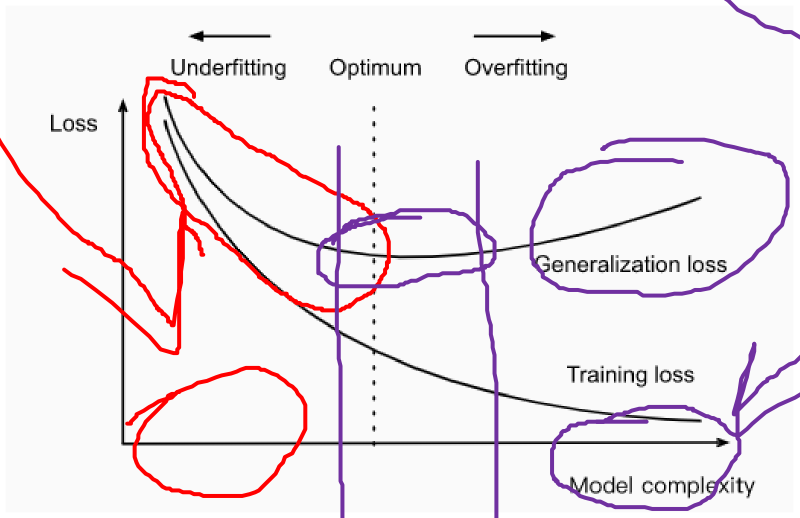
Types of errors

- **Estimation error (variance):** overfit
    - Caused due to small/noisy dataset
- **Structural error (bias):** underfit
    - Caused due to small set of predictors, F
- The two errors are cyclic, e.g. attempting to reduce structural errors with noisy data can lead to high estimation error.
- We need to find a balance between the two errors.

Under and Overfitting

$F =$





Model Selection

Overfitting. If model F is too big, then $f \in F$ performs

- well on training data, but poorly on test data.

Underfitting. If model F is too small, then $f \in F$ performs

- poorly on training data, and poorly on test data.

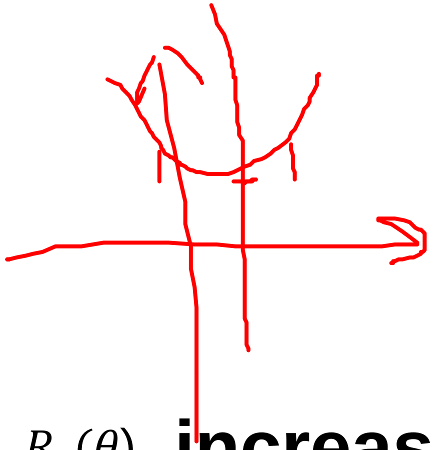
Finding a model with the right size is called model selection.

Optimization

Gradient Descent

- Use **gradient descent** to minimize $R_n(\theta)$

$$\nabla_\theta R_n(\theta) = \left[\frac{\partial R_n(\theta)}{\partial \theta_1}, \ldots, \frac{\partial R_n(\theta)}{\partial \theta_d}\right]^T$$



- Positive gradient points in the direction where $R_n(\theta)$ **increases**.
- Need to update the weight in the **opposite direction**.

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \nabla_\theta R_n(\theta)_{\theta=\theta^{(k)}}$$

Gradient Descent

- **Empirical Risk**

$$R_n(\theta) = \frac{1}{n}\sum_{t=1}^{n} \text{Loss}(y^{(t)} - \theta\cdot x^{(t)}) = \frac{1}{n}\sum_{t=1}^{n} (y^{(t)} - \theta\cdot x^{(t)})^2/2$$

- **Partial Derivative**

Num Num

$$\nabla_\theta (y^{(t)} - \theta\cdot x^{(t)})^2/2 = (y^{(t)} - \theta\cdot x^{(t)})\nabla_\theta (y^{(t)} - \theta\cdot x^{(t)}) = -(y^{(t)} - \theta\cdot x^{(t)})x^{(t)}$$

x^t

x_1^t

$\vdots$

x_d^t

- **Update of weight**

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \nabla_\theta R_n(\theta)_{\theta=\theta^{(k)}}$$

Update rule for batch gradient descent

$$\theta^{(k+1)} = \theta^{(k)} + \eta_k (y^{(t)} - \theta\cdot x^{(t)})x^{(t)}$$

Update rule for stochastic gradient descent

$k+1$

Stochastic Gradient Descent

1. Initialize the **weight** ($\theta^{(0)} = 0$).
2. Select $t \in \{1, \ldots, n\}$ at random

$$\theta^{(k+1)} = \theta^{(k)} + \eta_k (y^{(t)} - \theta \cdot x^{(t)}) x^{(t)}$$

3.Repeat Step (2) until stopping criterion is met.
(e.g. when improvement in $R_n(\theta)$ is small enough)

Closed Form Solution

- Minimize empirical risk directly by setting gradient to zero.

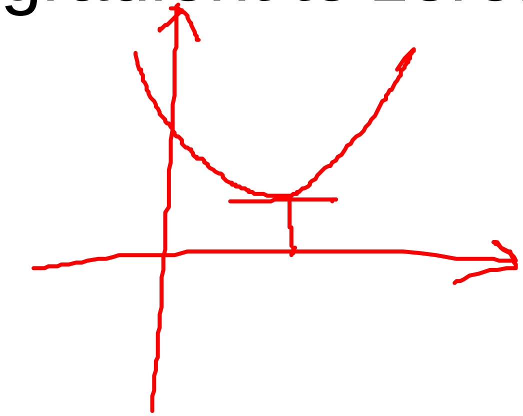
$$\nabla R_n(\theta)_{\theta=\hat{\theta}} = \frac{1}{n}\sum_{t=1}^{n} \nabla_\theta \left\{ (y^{(t)} - \theta \cdot x^{(t)})^2/2 \right\}_{|\theta=\hat{\theta}}$$

$$= \frac{1}{n}\sum_{t=1}^{n} \{-(y^{(t)} - \hat{\theta} \cdot x^{(t)})x^{(t)}\}$$

$$= -\frac{1}{n}\sum_{t=1}^{n} y^{(t)}x^{(t)} + \frac{1}{n}\sum_{t=1}^{n} (\hat{\theta} \cdot x^{(t)})x^{(t)}$$

$$= -\underbrace{\frac{1}{n}\sum_{t=1}^{n} y^{(t)}x^{(t)}}_{=b} + \underbrace{\frac{1}{n}\sum_{t=1}^{n} x^{(t)}(x^{(t)})^T}_{=A}\hat{\theta}$$

$$= -b + A\hat{\theta} = 0$$



Closed Form Solution

- If A is invertible ($n \geq d$), we can find the weight as below

$$\hat{\theta} = \boxed{A^{-1}} b.$$

$\propto d^3$

x^d

$n \geq d$

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & \\ & & & \\ x_1^n & x_2^n & \cdots & \end{bmatrix}$$

- Where, $b = \frac{1}{n} X^T \vec{y}$, $A = \frac{1}{n} X^T X$

Regularization

Ridge Regression

Temp. on Mars

Weight Age

Height $y \approx \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$

For simplicity, we ignore θ_0.

How do we ensure that $\theta_i = 0$ when feature x_i is irrelevant?

Pick simplest model that explains data ➔ **generalization**

Ridge Regression

Add a penalty.

$$J_{n,\lambda}(\theta) = \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_n} \frac{1}{2} (y - \theta^{\top} x)^2 + \frac{\lambda}{2} \|\theta\|^2$$

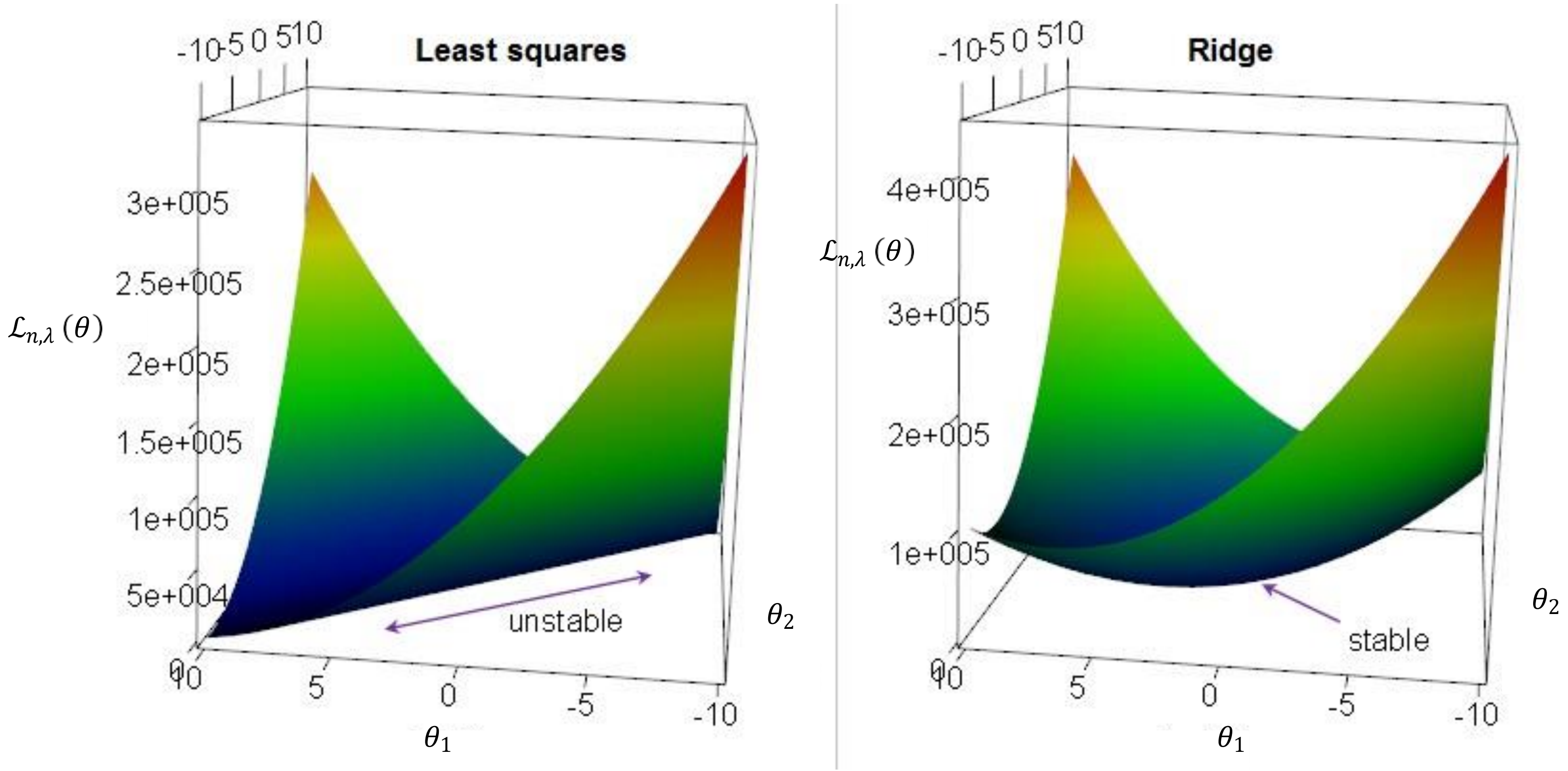
Pressure to fit data

Pressure to simplify model

Regularization parameter $\lambda \geq 0$

Regularizer

Ridge Regression



Training Algorithms

Ridge Regression $J_{n,\lambda}(\theta) = \frac{1}{n}\sigma_{(x,y)\in\mathcal{S}_n}\ \frac{1}{2}(y - \theta^{\top}x)^2 + \cancel{\frac{\lambda}{2}\|\theta\|^2}$

Gradient

$$\nabla J_{n,\lambda}(\theta) = \nabla_\theta \left\{ \frac{\lambda}{2}\|\theta\|^2 + (y^{(t)} - \theta \cdot x^{(t)})^2/2 \right\}_{|\theta=\theta^{(k)}}$$

$$\nabla J_{n,\lambda}(\theta) = \lambda\theta^{(k)} - (y^{(t)} - \theta^{(k)} \cdot x^{(t)})x^{(t)}$$

$\theta = \Theta^k$

Gradient Descent

$$\theta^{(k+1)} = \theta^k - \eta_k \nabla J_{n,\lambda}(\theta)$$

$$\theta^{(k+1)} = (1 - \lambda\eta_k)\theta^{(k)} + \eta_k(y^{(t)} - \theta \cdot x^{(t)})x^{(t)}$$

Without regulation, e.g., **λ = 0,** this shrinkage factor equals 1

Training Algorithms

Ridge Regression $J_{n,\lambda}(\theta) = \frac{1}{n} \sigma_{(x,y)\in\mathcal{S}_n} \frac{1}{2}(y - \theta^\top x)^2 + \frac{\lambda}{2}\|\theta\|^2$

Gradient $\nabla J_{n,\lambda}(\theta) = \lambda\theta + \frac{1}{n}(X^\top X)\theta - \frac{1}{n}X^\top Y$

Exact Solution

$$\nabla J_{n,\lambda}(\hat{\theta}) = 0 \quad \Leftrightarrow \quad \lambda\hat{\theta} + \frac{1}{n}(X^\top X)\,\hat{\theta} = \frac{1}{n}X^\top Y$$

$$\Leftrightarrow \quad \hat{\theta} = (n\lambda I + X^\top X)^{-1} X^\top Y$$

This matrix is always invertible when $\lambda > 0$.

Training Loss vs Test Loss

Training Loss

$$J_{n,\lambda}(\theta; \mathcal{S}_n) = \frac{1}{n} \sigma_{\boxed{(x,y)\in\mathcal{S}_n}} \frac{1}{2}(y - \theta^\top x)^2 + \boxed{\frac{\lambda}{2}\|\theta\|^2}$$

Test Loss/Error

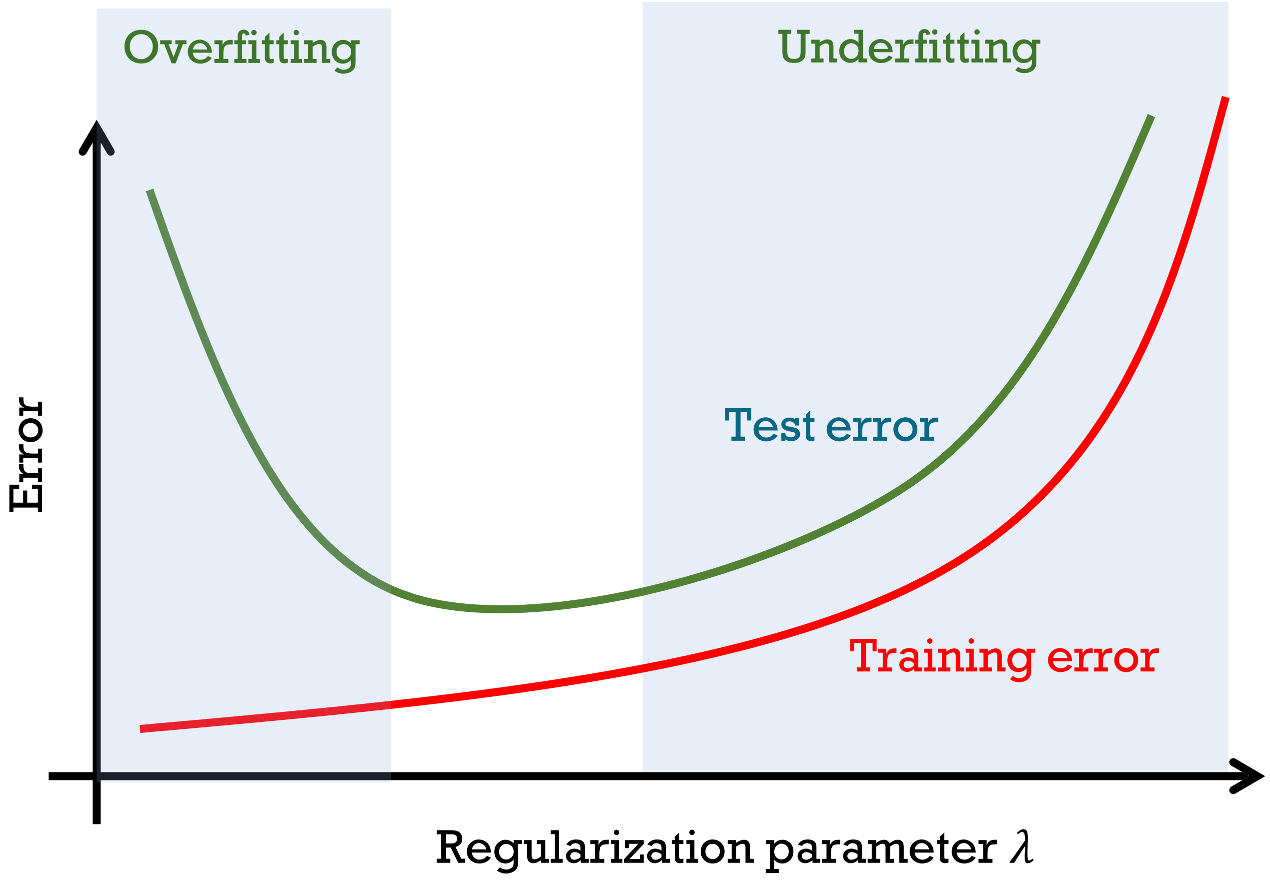
$$\mathcal{R}(\hat{\theta}; \mathcal{S}_{n'}) = \frac{1}{|\mathcal{S}_{n'}|} \sum_{\boxed{(x,y)\in\mathcal{S}_{n'}}} \frac{1}{2}\left(y - \hat{\theta}^\top x\right)^2$$

Training Error

$$\mathcal{R}(\hat{\theta}; \mathcal{S}_n) = \frac{1}{n} \sum_{\boxed{(x,y)\in\mathcal{S}_n}} \frac{1}{2}\left(y - \hat{\theta}^\top x\right)^2$$

The *training error* is the test loss applied to the training set, and it may be different from the training loss.

Effect of Regularization



Picking Hyperparameters

- The regularization parameter λ is an example of a *hyperparameter*, which affects the model complexity.

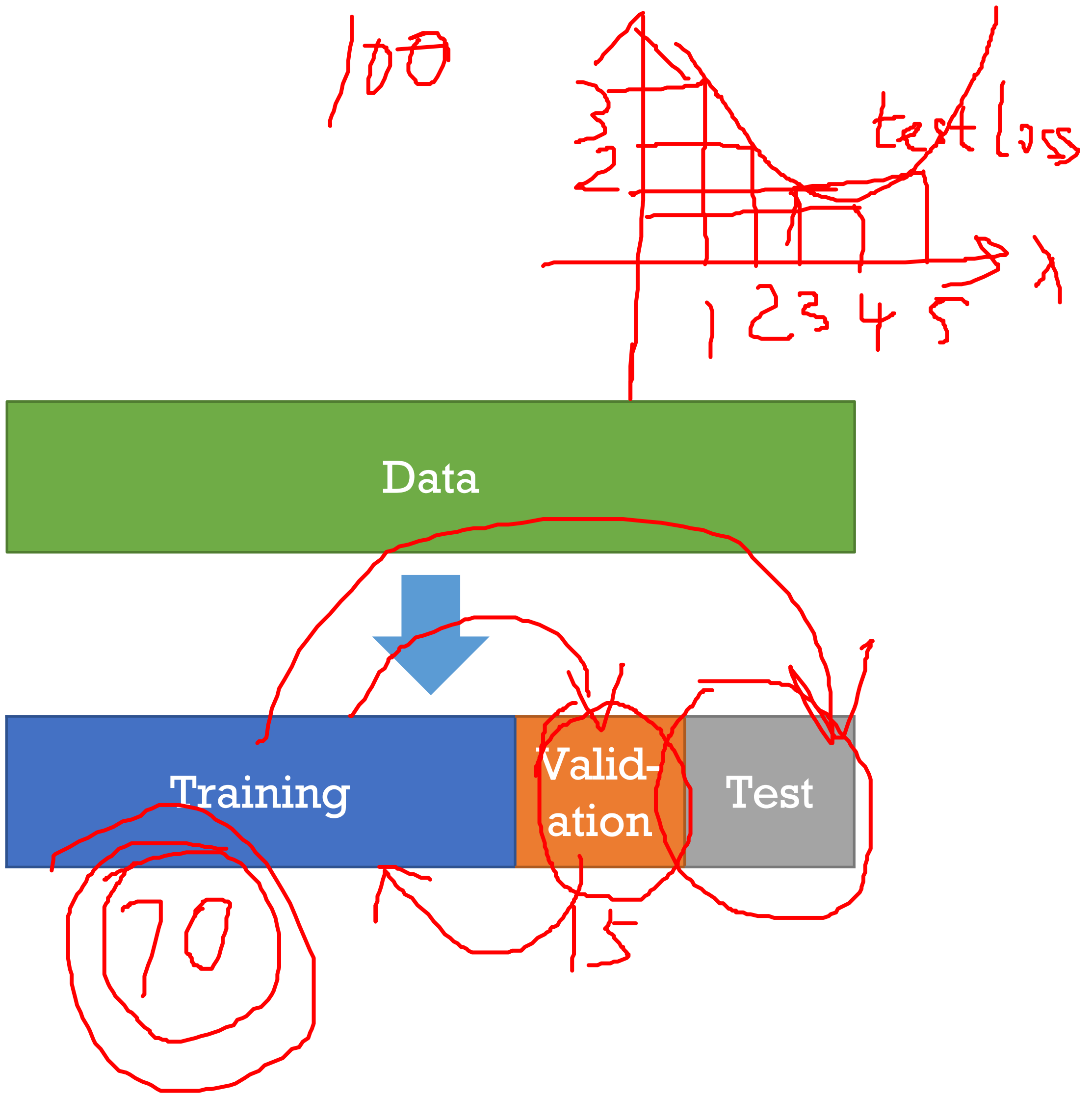
- We don't usually have access to the test data.
  How do we know if the value of λ minimizes the test loss?

- The solution is to create a *validation* data set, as a proxy
  to the test data, and to compute the *validation loss*.

Validation Set

Split the data into

- **Test set** $\mathcal{S}_{n'}$
  For evaluating, reporting performance at the end
- **Training set** $\mathcal{S}_n$
  For training optimal parameters in a model
- **Validation set** $\mathcal{S}_{\text{val}}$
  For model selection, e.g. picking λ in ridge regression. Acts as a proxy for test set.



Validation Loss

The *validation loss* is the test loss applied to the validation set.

Example. Ridge Regression

Test loss/error $\mathcal{R}(\theta\ ; \mathcal{S}_{n'}) = \frac{1}{|S_{n'}|} \sum_{(x,y)\in\mathcal{S}_{n'}} \frac{1}{2}\left(y - \theta^{\top}x\right)^2$

Validation loss/error $\mathcal{R}(\theta\ ; \mathcal{S}_{\mathrm{val}}) = \frac{1}{|s_{val}|} \sum_{(x,y)\in\mathcal{S}_{\mathrm{val}}} \frac{1}{2}\left(y - \theta^{\top}x\right)^2$

Model Selection