CS 540 Introduction to Artificial Intelligence
Linear Models & Linear Regression

University of Wisconsin-Madison

Fall, 2022

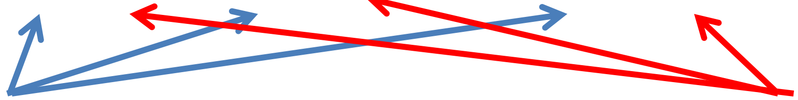
Outline

- Supervised Learning with Linear Models
  - Parameterized model, model classes, linear models, train vs. test
- Linear Regression
  - Least squares, normal equations, residuals, logistic regression

Supervised Learning

Supervised learning:

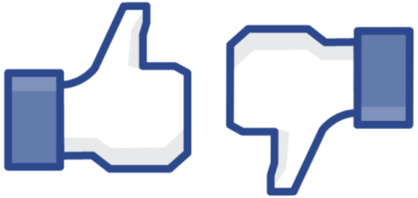
- Make predictions, classify data, perform regression
- Dataset: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$



Feature vector / Covariates / Input

Label

- Goal: find function $f : X \rightarrow Y$ to predict label on **new** data



Regression

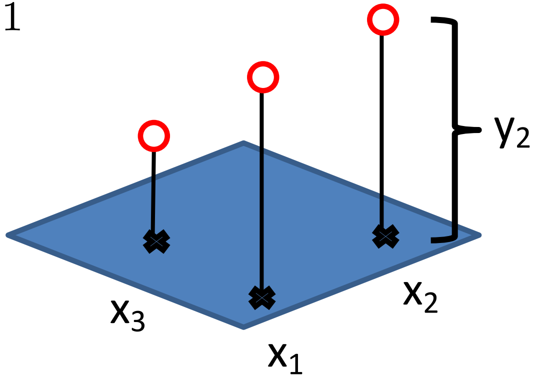
- Continuous label $y \in \mathbb{R}$
- Squared loss function $\ell(f(x), y) = (f(x) - y)^2$
- Finding f that minimizes the empirical risk

$$\frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$$

Functions/Models

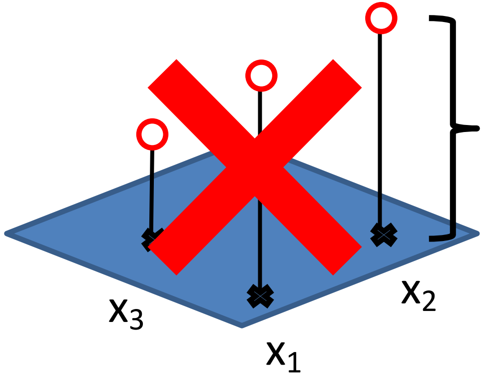
The function *f* is usually called a model

- Which possible functions should we consider?
- One option: **all functions**
  - Not a good choice. Consider $$f(x) = \sum_{i=1}^{n} 1\{x = x_i\} y_i$$
  - Training loss: **zero.** Can't do better!
  - How will it do on *x* not in the training set?



(cannot generalize)

Functions/Models



Don't want all functions

- Instead, pick a specific class
- Parametrize it by weights/parameters
- **Example:** linear models

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \theta_0 + x^T \theta$$

Weights/ Parameters

Training The Model

- Parametrize it by weights/parameters
- Minimize the loss

Best parameters = best function *f* →

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

Linear model class *f*

$$= \frac{1}{n} \sum_{i=1}^{n} \ell(\theta_0 + x_i^T \theta, y_i)$$

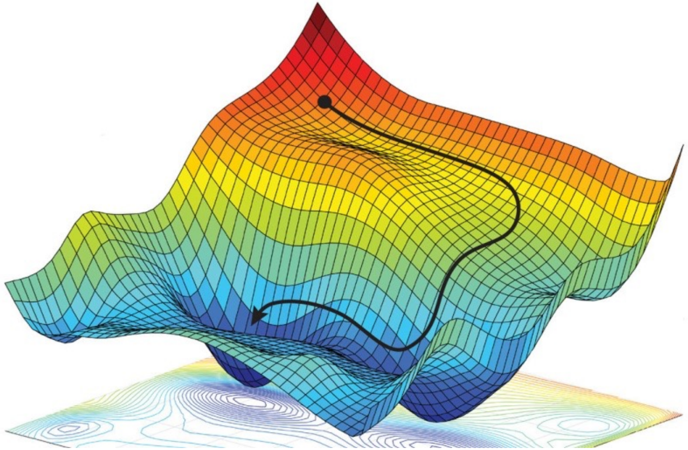
Square loss

$$= \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + x_i^T \theta - y_i)^2$$

How Do We Minimize?

- Need to solve something that looks like $\min_{\theta} g(\theta)$
- Generic optimization problem; many algorithms
  - A popular choice: **stochastic gradient descent (SGD)**

- Most algorithms iterative: find some sequence of points heading towards the optimum



M. Hutson

Train vs Test

Now we've trained, have some *f* parametrized by θ

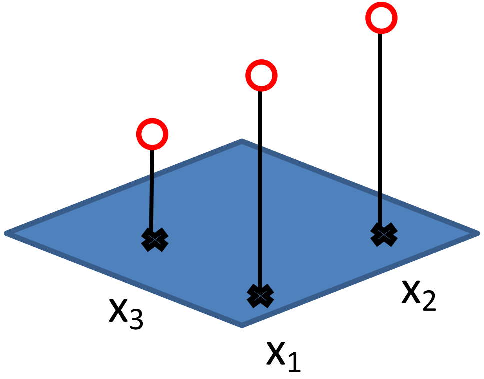
- Train loss is small → *f* predicts most x_i correctly
- How does *f* do on points not in training set? **"Generalizes!"**
- To evaluate this, reserve a **test** set. Do **not** train on it!

$$\underbrace{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)}_{\text{Training Data}} \quad \underbrace{(\mathbf{x}_{n+1}, y_{n+1}), \ldots, (\mathbf{x}_{n+p}, y_{n+p})}_{\text{Test Data}}$$

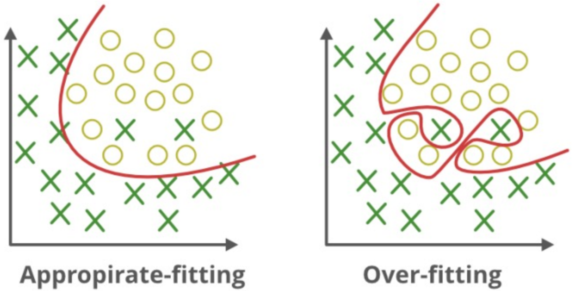
Train vs Test

Use the test set to evaluate *f*

- Why? Back to our "perfect" train function
- Training loss: 0. Every point matched perfectly
- How does it do on test set? **Fails completely**!



- Test set helps detect **overfitting**
  - Overfitting: too focused on train points
  - "Bigger" class: more prone to overfit
    - Need to consider **model capacity**



GFG

Break & Quiz

Q 2.1: When we train a model, we are

- A. Optimizing the parameters and keeping the features fixed.
- B. Optimizing the features and keeping the parameters fixed.
- C. Optimizing the parameters and the features.
- D. Keeping parameters and features fixed and changing the predictions.

Break & Quiz

Q 2.1: When we train a model, we are

- **A. Optimizing the parameters and keeping the features fixed.**
- B. Optimizing the features and keeping the parameters fixed.
- C. Optimizing the parameters and the features.
- D. Keeping parameters and features fixed and changing the predictions.

Break & Quiz

Q 2.1: When we train a model, we are

- **A. Optimizing the parameters and keeping the features fixed.**
- B. Optimizing the features and keeping the parameters fixed) (Feature vectors *x*i don't change during training).
- C. Optimizing the parameters and the features. (Same as B)
- D. Keeping parameters and features fixed and changing the predictions. (We can't train if we don't change the parameters)

Break & Quiz

- **Q 2.2**: You have trained a classifier, and you find there is significantly **higher** loss on the test set than the training set. What is likely the case?

- A. You have accidentally trained your classifier on the test set.
- B. Your classifier is generalizing well.
- C. Your classifier is generalizing poorly.
- D. Your classifier is ready for use.

Break & Quiz

- **Q 2.2**: You have trained a classifier, and you find there is significantly **higher** loss on the test set than the training set. What is likely the case?

- A. You have accidentally trained your classifier on the test set.
- B. Your classifier is generalizing well.
- **C. Your classifier is generalizing poorly.**
- D. Your classifier is ready for use.

Break & Quiz

- **Q 2.2**: You have trained a classifier, and you find there is significantly **higher** loss on the test set than the training set. What is likely the case?

- A. You have accidentally trained your classifier on the test set. **(No, this would make test loss lower)**
- B. Your classifier is generalizing well. **(No, test loss is high means poor generalization)**
- **C. Your classifier is generalizing poorly.**
- D. Your classifier is ready for use. **(No, will perform poorly on new data)**

Break & Quiz

- **Q 2.3**: You have trained a classifier, and you find there is significantly **lower** loss on the test set than the training set. What is likely the case?

- A. You have accidentally trained your classifier on the test set.
- B. Your classifier is generalizing well.
- C. Your classifier is generalizing poorly.
- D. Your classifier needs further training.

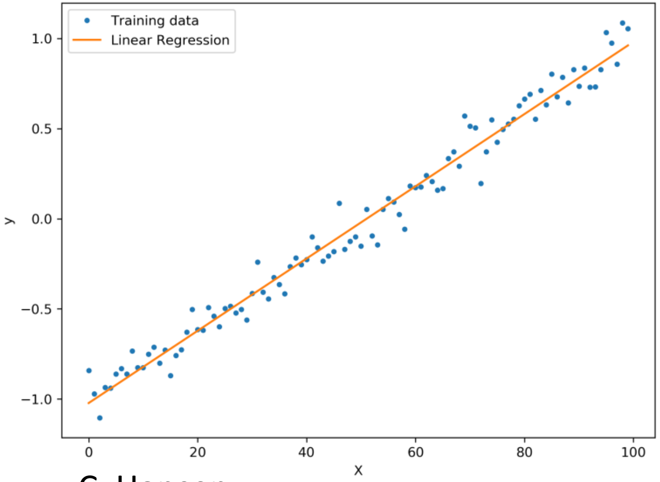
Break & Quiz

- **Q 2.3**: You have trained a classifier, and you find there is significantly **lower** loss on the test set than the training set. What is likely the case?

- **A. You have accidentally trained your classifier on the test set.** (This is very likely, loss will usually be the lowest on the data set on which a model has been trained)
- B. Your classifier is generalizing well.
- C. Your classifier is generalizing poorly.
- D. Your classifier needs further training.

Linear Regression

Simplest type of regression problem.



C. Hansen

- **Inputs**: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
  - *x*'s are vectors, *y*'s are scalars.
  - "**Linear**": predict a linear combination of x components + intercept

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \theta_0 + x^T \theta$$

- **Want**: parameters θ

Linear Regression Setup

Problem Setup

- Goal: figure out how to minimize square loss
- Let's organize it. Train set $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
  - Since $f(x) = \theta_0 + x^T\theta$, use a notational trick by augmenting feature vector with a constant dimension of 1:

$$x = \begin{bmatrix} 1 \\ x \end{bmatrix}$$

  - Then, with this one more dimension we can write (θ contains θ_0 now)

$$f(x) = x^T\theta$$

Linear Regression Setup

Problem Setup

- Train set $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
- Take train features and make it a n*(d+1) matrix, and y a vector:

$$X = \begin{bmatrix} x_1^T \\ \ldots \\ x_n^T \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ \ldots \\ y_n \end{bmatrix}$$

- Then, the empirical risk is $\frac{1}{n}\|X\theta - y\|^2$

Finding The Estimated Parameters

Have our loss: $\frac{1}{n}\|X\theta - y\|^2$

- Could optimize it with SGD, etc…
- But the minimum also has a closed-form solution (vector calculus):

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

Hat: indicates an estimate

Not always invertible…

“Normal Equations”

How Good are the Estimated Parameters?

Now we have parameters $\hat{\theta} = (X^T X)^{-1} X^T y$

- How good are they?
- Predictions are $f(x_i) = \hat{\theta}^T x_i = ((X^T X)^{-1} X^T y)^T x_i$
- Errors ("residuals")

$$|y_i - f(x_i)| = |y_i - \hat{\theta}^T x_i| = |y_i - ((X^T X)^{-1} X^T y)^T x_i|$$

- If data is linear, residuals are 0. Almost never the case!
- Mean squared error on a test set $\frac{1}{m} \sum_{i=n+1}^{n+m} (\hat{\theta}^T x_i - y_i)^2$

Linear Regression → Classification?

What if we want the same idea, but *y* is 0 or 1?

- Need to convert the $\theta^T x$ to a probability in [0,1]

$$p(y=1|x) = \frac{1}{1+\exp(-\theta^T x)}$$

← **Logistic function**

Why does this work?

- If $\theta^T x$ is really big, $\exp(-\theta^T x)$ is really small → *p* close to 1
- If really negative exp is huge → *p* close to 0

"Logistic Regression"

Reading

- Linear regression, logistic regression, stochastic gradient descent by Prof. Zhu https://pages.cs.wisc.edu/~jerryzhu/cs540/handouts/regression.pdf