CS 540 Introduction to Artificial Intelligence
Linear Regression & Linear Models

University of Wisconsin–Madison
Fall 2025, Section 3
September 29, 2025

Announcements

- HW3 due Friday 10/3 at 11:59 PM

- Class roadmap:

ML: Unsupervised Learning
ML Linear Regression
Machine Learning: K - Nearest Neighbors & Naive Bayes
Machine Learning: Neural Networks I (Perceptron)
Machine Learning: Neural Networks II

Supervised Learning

Supervised Learning

Supervised learning:

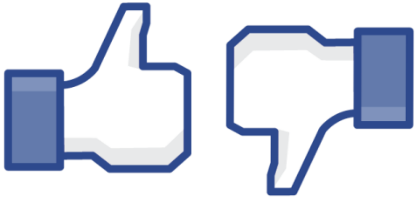
- Make predictions, classify data, perform regression
- Dataset: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$

Feature vector / Covariates / Input

Label

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

Select a **model** f from a class of possible models



indoor

outdoor

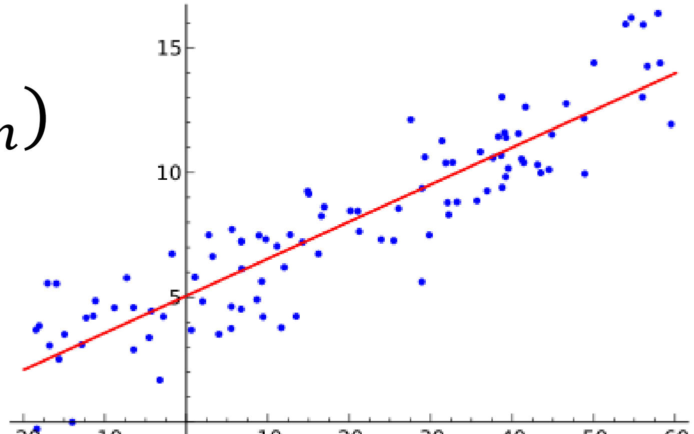
Linear Regression

- Simplest form of regression
- Find a line that fits the data

- Example:
  - Training data $(x_1, y_1), \dots, (x_n, y_n)$
  - Find $a, b \in \mathbb{R}$ such that

$$\forall i,\ y_i \approx a x_i + b$$

scalars



Basic Recipe for Supervised Learning

1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

Usually: apply a variant of **gradient descent**

Usually: compute loss on **test set**

Basic Recipe: Linear Regression

1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

1. Linear functions
2. Squared error
3. Gradient descent
4. Test/train split

1) Model Class for Linear Regression

- All linear functions from features to labels
  - Features $x_i \in \mathbb{R}^d$
  - Labels $y_i \in \mathbb{R}$
  - Models: $f_\theta(x_i) = \langle \underbrace{\theta}_{\theta \in \mathbb{R}^d}, x_i \rangle$
- Model class **parameterized** by $\theta \in \mathbb{R}^d$

Basic Recipe
1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

Earlier example:
$f_{(a,b)}(x_i) = ax_i + b$

Aside: Notational Trick

- When x is a scalar: $f_{(a,b)}(x) = ax + b$
- When x is a vector:

$$f_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$$

- Give x a "dummy dimension" to simplify notation

Old

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

New

$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

$$f_\theta(x) = \langle \theta, x \rangle = [\theta_0 \quad \theta_1 \quad \theta_2 \quad \cdots \quad \theta_d] \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

1) Model Class for Linear Regression

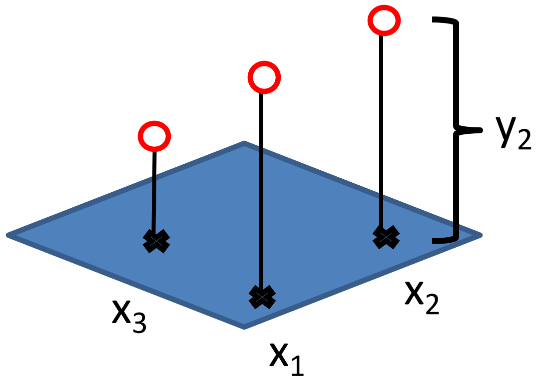
- All linear functions from features to labels
  - Features $x_i \in \mathbb{R}^d$
  - Labels $y_i \in \mathbb{R}$
  - Models: $f_\theta(x_i) = \langle \underset{\theta \in \mathbb{R}^d}{\theta}, x_i \rangle$
- Model class **parameterized** by $\theta \in \mathbb{R}^d$

Basic Recipe
1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

Why restrict the model class?

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

- Want functions that **generalize**



2) Loss Function for Linear Regression

- **Loss** measures quality of prediction
- We use **squared error**:

$$(f_\theta(x) - y)^2$$

- Loss on a single data point

$$\ell(\theta; x_i, y_i) = (f_\theta(x_i) - y_i)^2 = (\langle \theta, x_i \rangle - y_i)^2$$

- Loss on a dataset

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

"Mean squared error"

Basic Recipe
1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

3) Optimizing parameters for linear regression

- Find a line that fits the data

$$\min_\theta \frac{1}{n}\sum_{i=1}^{n}(\langle\theta, x_i\rangle - y_i)^2$$

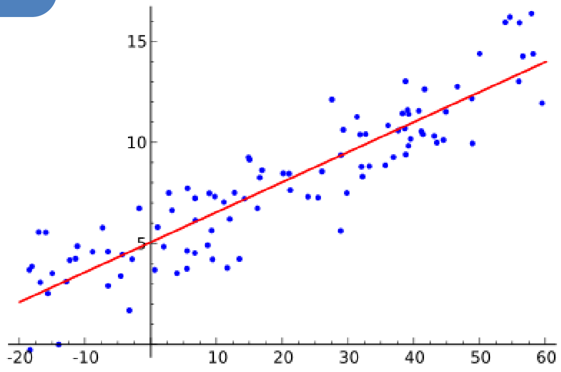
- In general:

$$\min_\theta L(\theta; X, y)$$

"Empirical risk minimization"

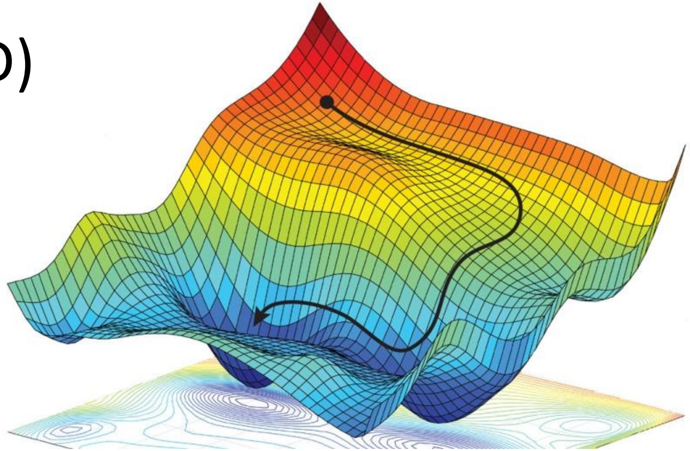
- How can we solve this?

Basic Recipe
1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output



How Do We Optimize Parameters?

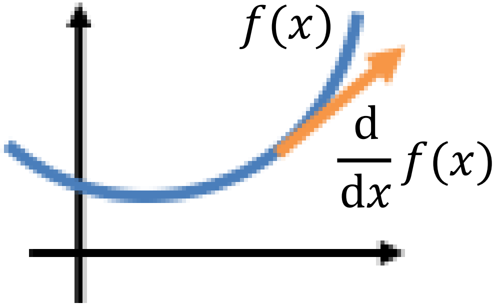
- Need to solve something that looks like $\min_{\theta} g(\theta)$
- Generic optimization problem; many algorithms
- Most popular: variants of **gradient descent**
  - Stochastic Gradient Descent (SGD)
  - Momentum
  - Adam



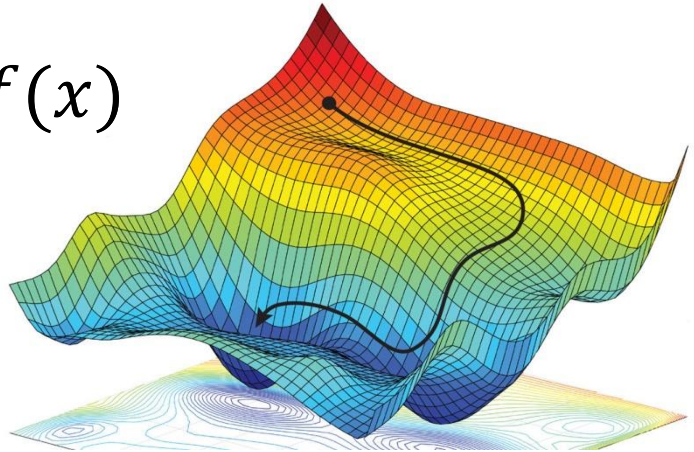
M. Hutson

Gradients & Gradient Descent

- One dimension: derivative $\frac{\mathrm{d}}{\mathrm{d}x} f(x)$
  - How to shift x to make $f(x)$ larger



- Higher dimensions: gradient $\nabla f(x)$
  - Direction where f grows fastest

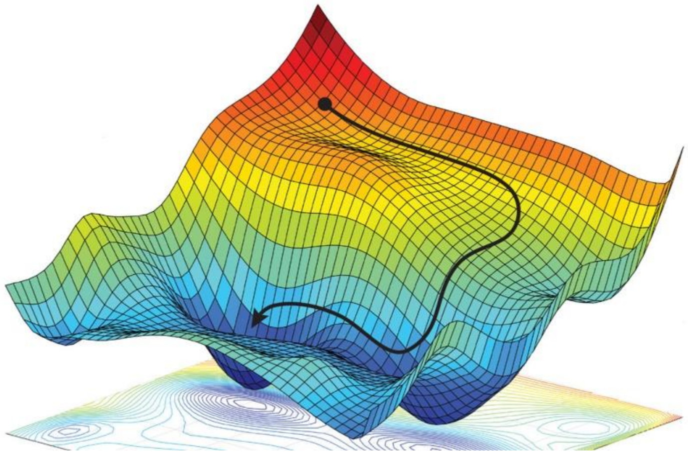


M. Hutson

Gradients & Gradient Descent

- Gradient descent takes iterative steps to make loss function smaller

Gradient Descent
<u>Input:</u> dataset (X, y), loss function L, number of steps T, step size η
1. Initialize θ_0 2. For $t = 1,2,\ldots,T$ 3. Calculate $g_t = \nabla L(\theta_{t-1}; X, y)$ 4. Update $\theta_t \leftarrow \theta_{t-1} - \eta g_t$ 5. Return θ_T



M. Hutson

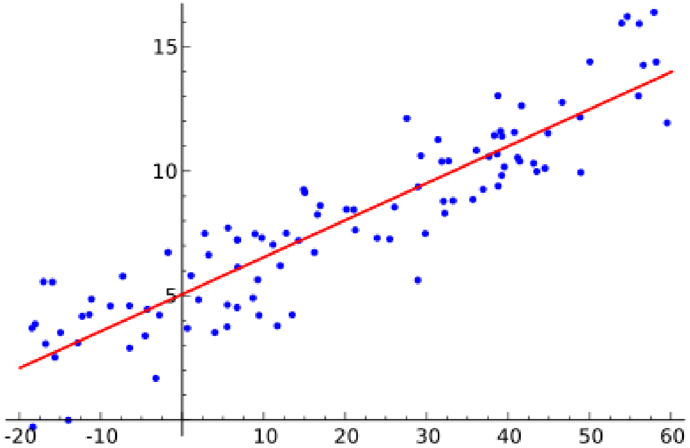
3) Optimizing parameters for linear regression

Basic Recipe
1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

- Find a line that fits the data

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (\langle \theta, x_i \rangle - y_i)^2$$

- Apply gradient descent to find θ^* with smallest squared loss



Aside: A Closed-Form Solution

- Find a line that fits the data

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (\langle \theta, x_i \rangle - y_i)^2$$

- Vector calculus leads to a closed-form solution (i.e., "take the derivative and set to 0")

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

"normal equations"

Special fact for linear regression

4) Evaluating the model we found

Basic Recipe
1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

- We found some θ^* with low loss
  - I.e., accurate labels for training data
- Is it useful?

- Recall the goal of supervised learning:
  find function $f: X \rightarrow Y$ to predict label on **new** data

4) Evaluation & Train/Test Split

- Reserve a **test set**
- Do not train on it!

Basic Recipe

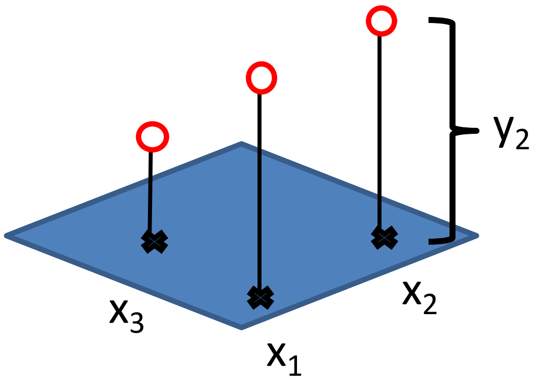
1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

$$\underbrace{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)}_{\text{Training Data}} \qquad \underbrace{(x_{n+1}, y_{n+1}), (x_{n+2}, y_{n+2}), \dots, (x_{n+k}, y_{n+k})}_{\text{Test Data}}$$

We get concerned if: $L(\theta^*; X_{\text{test}}, y_{\text{test}}) \gg L(\theta^*; X_{\text{train}}, y_{\text{train}})$

Test set and generalization

- Recall our function that fits the training set perfectly
  - $f(x) = \sum_{i=1}^{n} 1\{x = x_i\} \cdot y_i$



- Error on test set will be large!

- We want functions that **generalize**

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"

Basic Recipe: Linear Regression

1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

1. Linear functions
2. Squared error
3. Gradient descent
4. Test/train split

Reading

- Russell & Norvig, Sections 4.2 & 19.6
- Linear regression, logistic regression, stochastic gradient descent by Prof. Zhu https://pages.cs.wisc.edu/~jerryzhu/cs540/handouts/regression.pdf