



# CS 540 Introduction to Artificial Intelligence

## **Linear Models & Linear Regression**

Fred Sala  
University of Wisconsin-Madison

Feb 25, 2021

# Announcements

- **Homeworks:**
  - HW4: you should start if you haven't!
- **Midterm** coming up on March 17<sup>th</sup>
  - More info soon.

- **Class roadmap:**

Thursday, Feb 23	ML Unsupervised II
<b>Thursday, Feb 25</b>	<b>ML Linear Regression</b>
Tuesday, March 2	ML: Naïve Bayes, KNN
Thursday, March 4	ML: Neural Networks I

Machine Learning

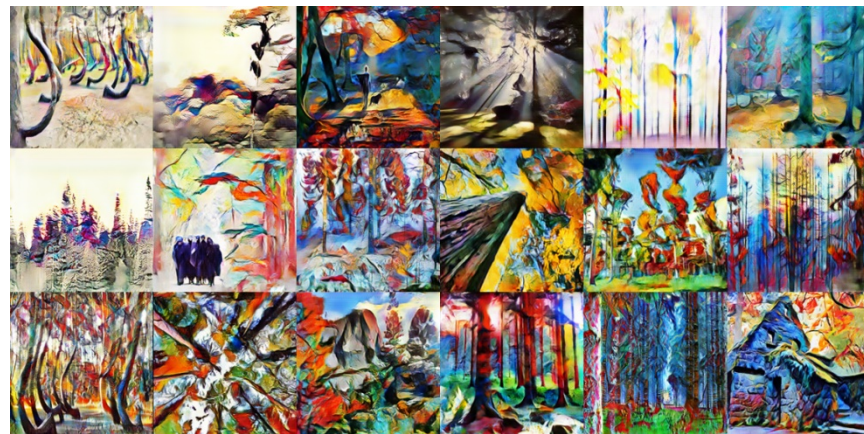
# Outline

- Unsupervised Learning: Density Estimation
  - Kernel density estimation: high-level intro
- Supervised Learning & Linear Models
  - Parameterized model, model classes, linear models, train vs. test
- Linear Regression
  - Least squares, normal equations, residuals, logistic regression

# Short Intro to Density Estimation

Goal: given samples  $x_1, \dots, x_n$  from some distribution  $P$ , estimate  $P$ .

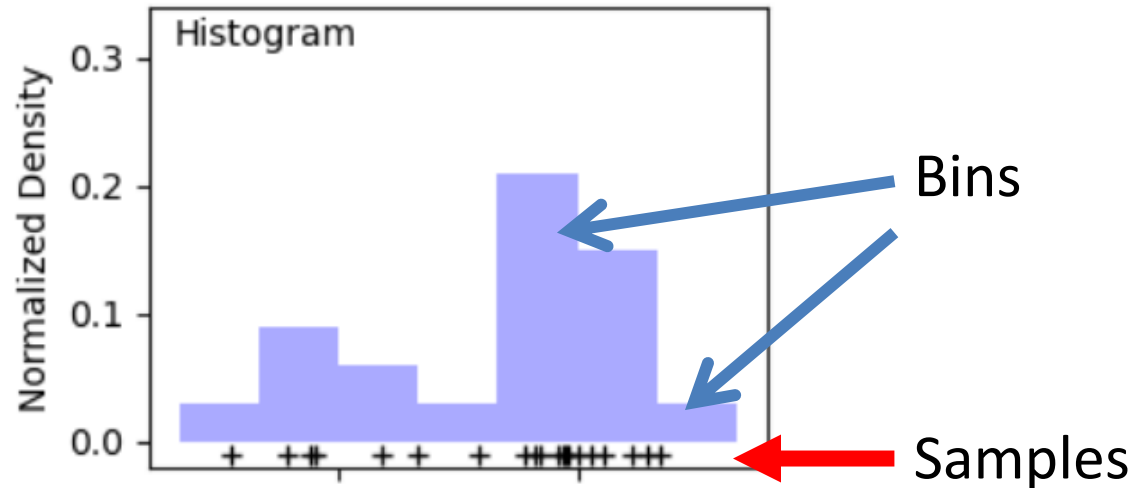
- Compute statistics (mean, variance)
- Generate samples from  $P$
- Run inference



Zach Monge

# Simplest Idea: Histograms

Goal: given samples  $x_1, \dots, x_n$  from some distribution  $P$ , estimate  $P$ .



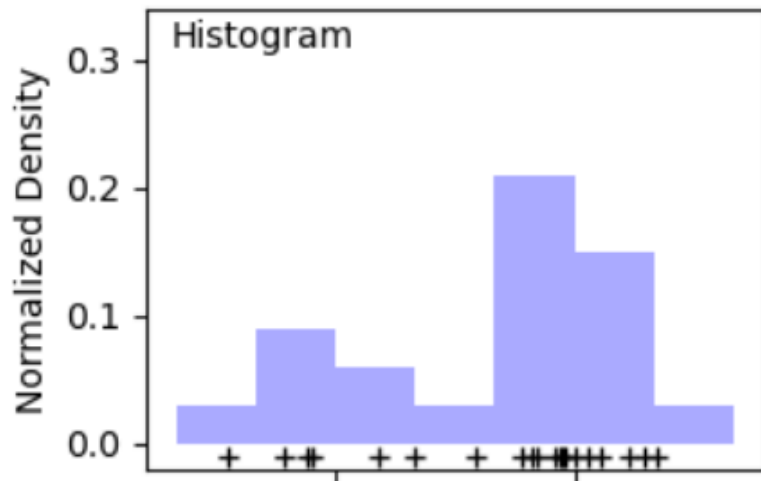
Define bins; count # of samples in each bin, normalize

# Simplest Idea: Histograms

Goal: given samples  $x_1, \dots, x_n$  from some distribution  $P$ , estimate  $P$ .

## Downsides:

- i) High-dimensions: most bins empty
- ii) Not continuous
- iii) How to choose bins?



# Kernel Density Estimation

Goal: given samples  $x_1, \dots, x_n$  from some distribution  $P$ , estimate  $P$ .

**Idea:** represent density as combination of “kernels”

$$f(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

Center at each point

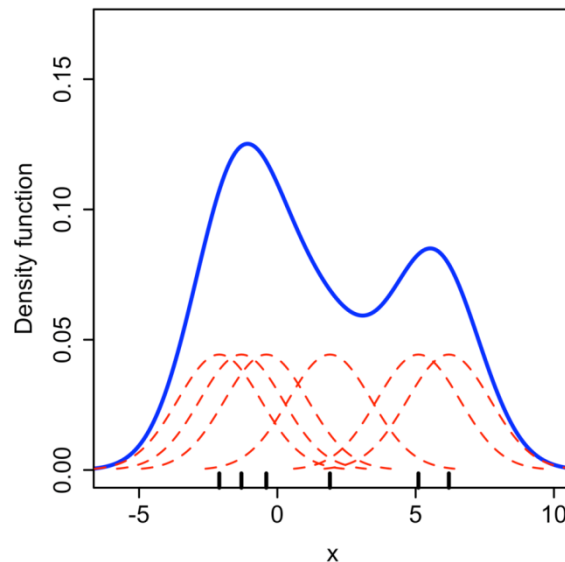
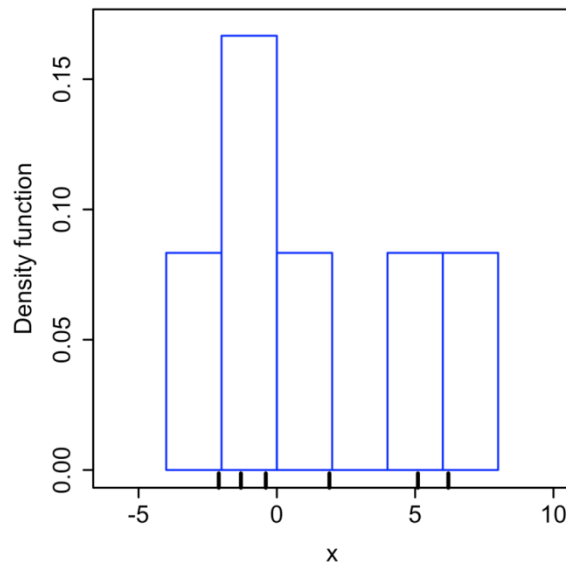
Kernel function: often Gaussian

Width parameter

# Kernel Density Estimation

**Idea:** represent density as combination of kernels

- “Smooth” out the histogram





# Back to Supervised Learning

## Supervised learning:

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



Features / Covariates / Input

Labels / Outputs

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



indoor



outdoor

# Back to Supervised Learning

How do we know a function  $f$  is good?

- Intuitively: “matches” the dataset  $f(x_i) \approx y_i$
- More concrete: pick a **loss function** to measure this:  $\ell(f(x), y)$
- Training loss/empirical loss/empirical risk

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

 Loss / Cost / Objective  
Function

- Find a  $f$  that minimizes the loss on the training data (ERM)

# Loss Functions

What should the loss look like?

- If  $f(x_i) \approx y_i$  , should be small (0 if equal!)
- For classification: 0/1 loss  $\ell(f(x), y) = 1\{f(x_i) \neq y_i\}$
- For regression, square loss  $\ell(f(x), y) = (f(x_i) - y_i)^2$

Others too! We'll see more.

# Functions/Models

The function  $f$  is usually called a model

- Which possible functions should we consider?

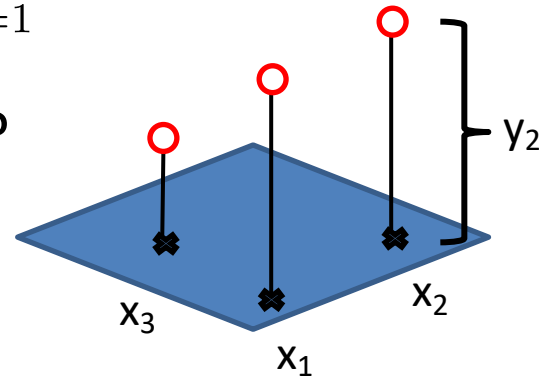
- One option: **all functions**

- 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?



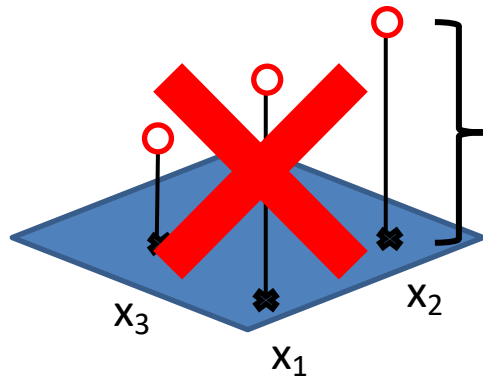
# Functions/Models

Don't want all functions

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


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

Weights/ Parameters





# Training The Model

- Parametrize it by weights/parameters
- Minimize the loss

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

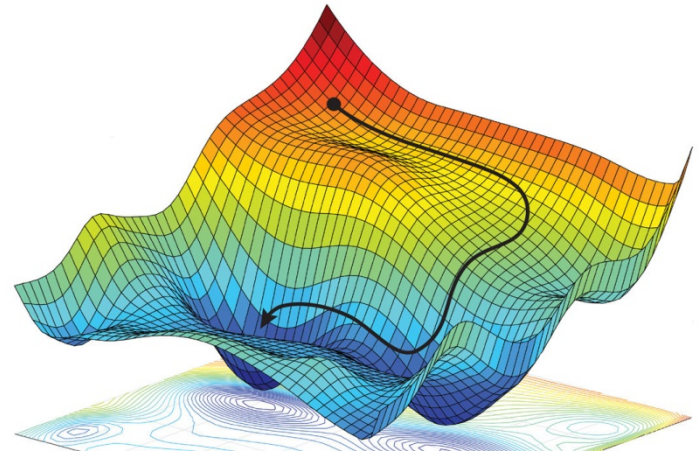
best function  $f$

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

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

# 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



# Train vs Test

Now we've trained, have some  $f$  parametrized by  $\theta$

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

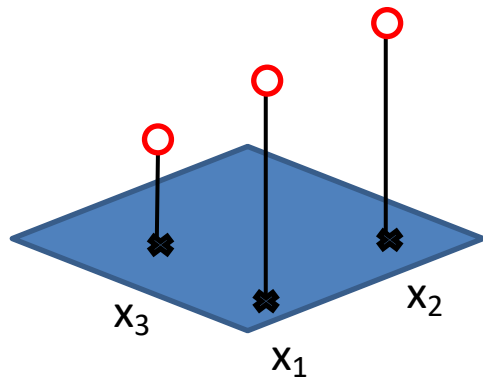
$$\underbrace{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)}_{\text{Training Data}} \quad \underbrace{(\mathbf{x}_{n+1}, y_{n+1}), \dots, (\mathbf{x}_{n+p}, y_{n+p})}_{\text{Test Data}}$$



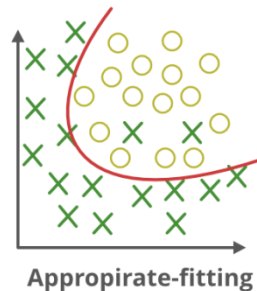
# Train vs Test

Use the test set to evaluate  $f$

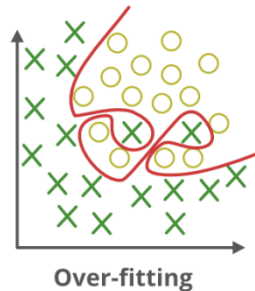
- Why? Back to our “perfect” train function
- Training loss: 0. Every point matched perfectly
- How does it do on test set? **Fails completely!**



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



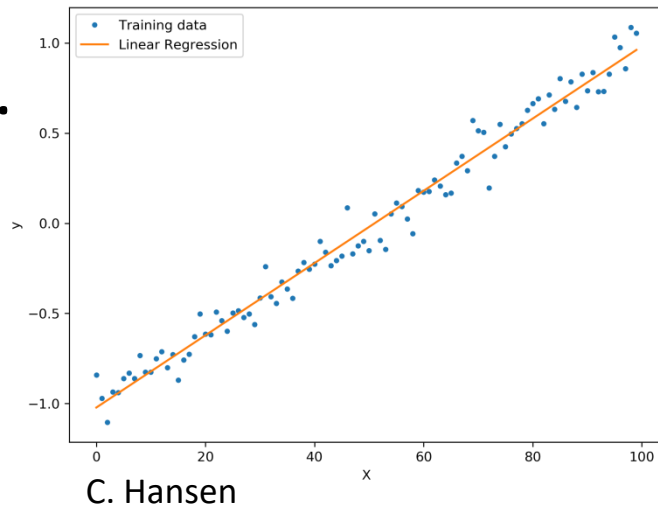
GFG



# Linear Regression

Simplest type of regression problem.

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



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

- **Want:** parameters  $\theta$

# Linear Regression Setup

## Problem Setup


- Goal: figure out how to minimize square loss
- Let's organize it. Train set  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$ 
  - Since  $f(x) = \theta_0 + x^T \theta$ , wrap intercept:  $f(x) = x^T \theta$
  - Take train data and make it a matrix/vector:
  - Then, square loss is

$$\frac{1}{n} \sum_{i=1}^n (x_i^T \theta - y_i)^2 = \frac{1}{n} \|X^T \theta - y\|^2$$
$$X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

# Finding The Optimal Parameters

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

- Could optimize it with SGD, etc...
- No need: minimum has a solution (easy with vector calculus)

Hat: indicates an estimate   $\hat{\theta} = (X^T X)^{-1} X^T y$

 Not always invertible...

**“Normal Equations”**

# How Good are the Optimal 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!

# Train/Test for Linear Regression?

So far, residuals measure error on **train** set

- Sometimes that's all we care about (**Fixed Design LR**)
  - Data is deterministic.
  - Goal: find best linear relationship on dataset
- Or, create a test set and check (**Random Design LR**)
  - Common: assume data is  $y = \theta^T x + \varepsilon$
  - The more noise, the less linear



0-mean  
Gaussian noise

# 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  $[0,1]$

$$p(y = 1|x) = \frac{1}{1 + \exp(-\theta^T x)} \quad \leftarrow \text{Logistic function}$$

Why does this work?

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

**“Logistic Regression”**