



CS 540 Introduction to Artificial Intelligence

Linear Models & Linear Regression

Yingyu Liang
University of Wisconsin-Madison
Oct 12, 2021

Based on slides by Fred Sala

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 .

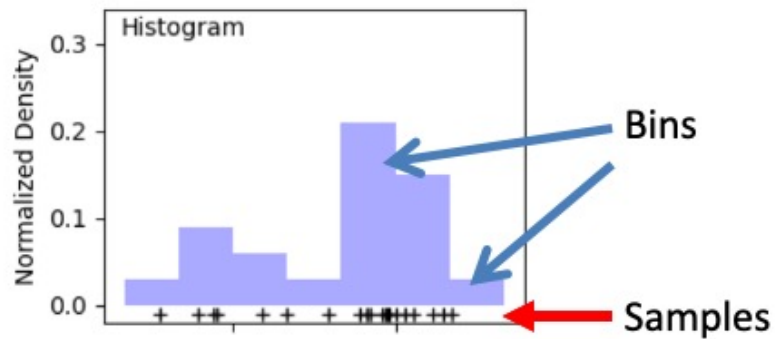
- Compute statistics (mean, variance)
- Generate samples from P
- Run inference



Zach Monge

Simplest Idea: Histograms

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



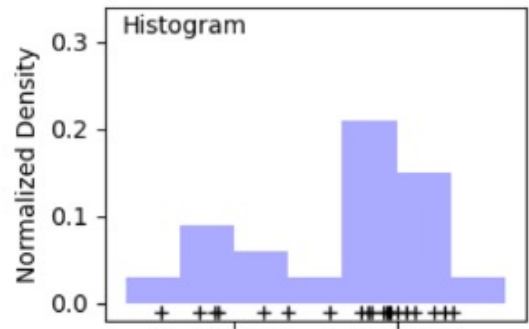
Define bins; count # of samples in each bin, normalize

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

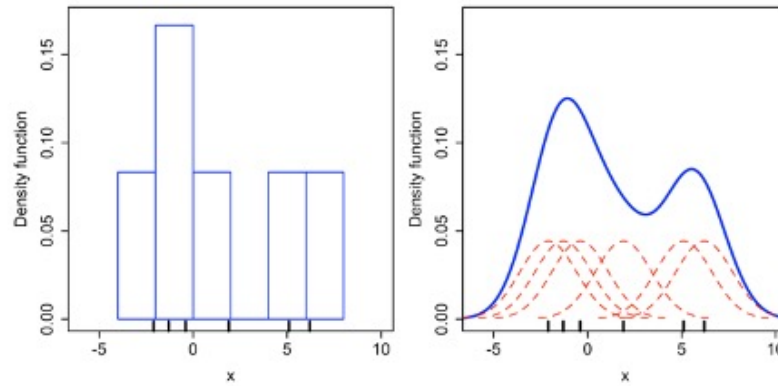
Width parameter

The diagram shows the kernel density estimation formula $f(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$. A red arrow points from the text 'Center at each point' to the x_i term in the denominator of the kernel function. A green arrow points from the text 'Kernel function: often Gaussian' to the K term. A blue arrow points from the text 'Width parameter' to the h term in the denominator of the kernel function.

Kernel Density Estimation

Idea: represent density as combination of kernels

- “Smooth” out the histogram



Break & Quiz

Q 1.1: Which of the following is not true?

- A. Using a Gaussian kernel for KDE, all possible values for x_i will have non-zero probability.
- B. The goal of KDE is to approximate the true probability distribution function of X .
- C. When using a histogram, every bucket must be represented explicitly in memory
- D. With some kernels, KDE can assign zero probability to some subset of values for x_i .

Break & Quiz

Q 1.1: Which of the following is not true?

- A. Using a Gaussian kernel for KDE, all possible values for x_i will have non-zero probability.
- B. The goal of KDE is to approximate the true probability distribution function of X .
- **C. When using a histogram, every bucket must be represented explicitly in memory**
- D. With some kernels, KDE can assign zero probability to some subset of values for x_i .

Break & Quiz

Q 1.1: Which of the following is not true?

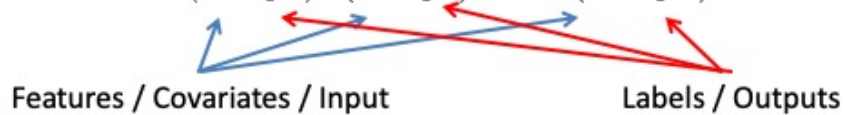
- A. Using a Gaussian kernel for KDE, all possible values for x_i will have non-zero probability. (Gaussian PDF positive for all inputs)
- B. The goal of KDE is to approximate the true probability distribution function of X . (same goal as histograms)
- **C. When using a histogram, every bucket must be represented explicitly in memory**
- D. With some kernels, KDE can assign zero probability to some subset of values for x_i . (Consider $K = \text{uniform}(0,1)$)

About C: we don't necessarily need to keep every bucket explicitly. For example, we can keep only those that are non-empty.

Back to Supervised Learning

Supervised learning:

- Make predictions, classify data, perform regression
- Dataset: $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$



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



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)

Three key elements:

1. A class of functions from which we choose our model
2. A loss function measuring the match between the prediction and the true label
3. A method to minimize the training loss

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 $\ell(f(x), y) = \mathbb{1}\{f(x_i) \neq y_i\}$
- For regression, square loss $\ell(f(x), y) = (f(x_i) - y_i)^2$

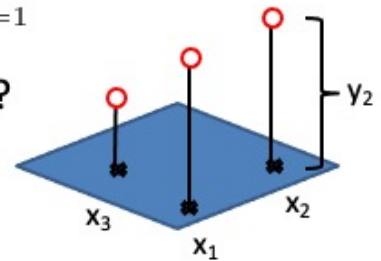
Others too! We'll see more.

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?



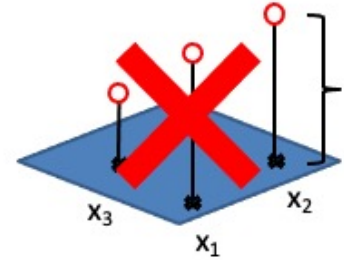
If we consider choosing our model among the family of all (measurable) functions, then we can have the constructed function which has zero training loss but can have large error on future test inputs.

The phenomenon that a function gets much lower training loss than on test inputs is called overfitting. It usually happens when we choose model from a too large family of functions. Intuitively, it's likely that a very large family of functions contains some function that can fit the training data but not the test data.

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


So we should choose a more specific class of functions. This lecture considers the family of linear functions.

Training The Model

- Parametrize it by weights/parameters
- Minimize the loss

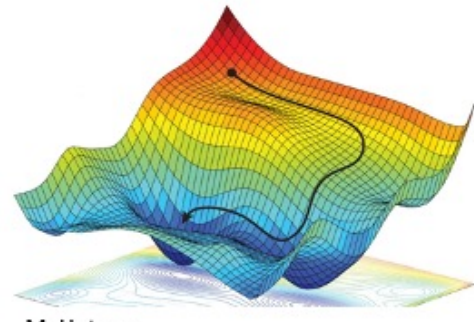
Best parameters = best function f  $\min_{\theta} \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i)$

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



M. Hutson

We have specify our class of functions and the loss function. Now we need a method to do the optimization. Many modern machine learning methods use SGD to do the optimization.

Train vs Test

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

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

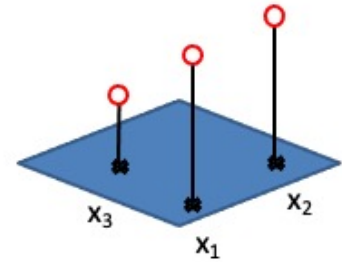
$$\underbrace{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)}_{\text{Training Data}} \quad \underbrace{(\mathbf{x}_{n+1}, y_{n+1}), \dots, (\mathbf{x}_{n+p}, y_{n+p})}_{\text{Test Data}}$$

We say a model generalizes well, if it gets good predictions on test data.

Train vs Test

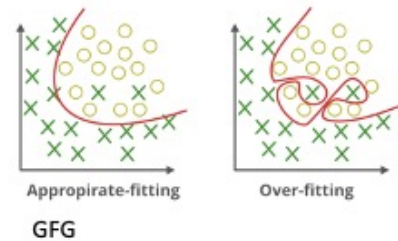
Use the test set to evaluate f

- 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**



What we really care about is the performance of the model on future test data. We should not use the performance on the training data to evaluate the quality of the model (because there could be overfitting).

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.

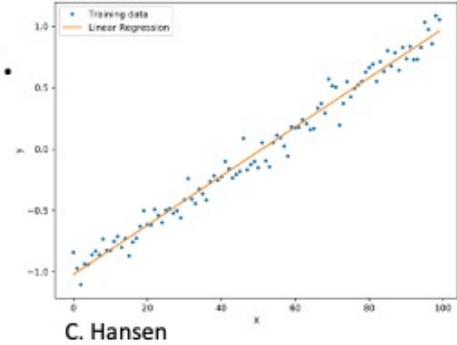
Linear Regression

Simplest type of regression problem.

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



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: $X = [x_1 \ x_2 \ \dots \ x_n]$
 - Then, square loss is

$$\frac{1}{n} \sum_{i=1}^n (x_i^T \theta - y_i)^2 = \frac{1}{n} \|X^T \theta - y\|^2$$

Usually, we wrap intercept by doing:

1. Let $\bar{\theta}$ be the concatenation of $[\theta_0 \ \theta]$
2. Let \bar{x} be the concatenation of $[1 \ x]$.


Then $f(x) = \langle \bar{x}, \bar{\theta} \rangle$. For simplicity, we reload the notation (writing \bar{x} as x and $\bar{\theta}$ as θ), and then write $f(x) = \langle x, \theta \rangle$.


X : each column is a data point.

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”

Can do SGD.

When $X^T X$ is invertible we can get a closed form solution.

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 → p close to 1
- If really negative exp is huge → p close to 0

“Logistic Regression”

For binary classification with labels $\{0,1\}$, we can squash the linear function output to a probability value in $[0,1]$ and use that to model $p(y=1|x)$.