CS 540 Introduction to Artificial Intelligence
Linear Models & Linear Regression
Yudong Chen
University of Wisconsin-Madison
Oct 12, 2021
Announcements

• **Homeworks:**
  – HW5: due next Tuesday

• **Class roadmap:**

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thursday, Oct 7</td>
<td>ML Unsupervised II</td>
</tr>
<tr>
<td><strong>Tuesday, Oct 12</strong></td>
<td><strong>ML Linear Regression</strong></td>
</tr>
<tr>
<td>Thursday, Oct 14</td>
<td>ML: Naïve Bayes, KNN</td>
</tr>
<tr>
<td>Tuesday, Oct 19</td>
<td>ML: Neural Networks I</td>
</tr>
</tbody>
</table>
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 $x_1, \ldots, x_n$ from some distribution $P$, estimate $P$.

• Compute statistics (mean, variance)
• Generate samples from $P$
• Run inference
Simplest Idea: Histograms

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

Define bins; count # of samples in each bin, normalize
Simplest Idea: Histograms

Goal: given samples $x_1, \ldots, x_n$ 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 $x_1, ..., x_n$ 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

![Kernel Density Estimation Diagram](image-url)
Q 1.1: Which of the following is not true?

- A. Using a Gaussian kernel for KDE, all possible values for $x$ will have non-zero probability $f(x)$.
- B. The goal of KDE is to approximate the true probability distribution function of $x$.
- C. KDE cannot be applied if the data $x_1, \ldots, x_n$ are vectors.
- D. With some kernels, KDE can assign zero probability to some subset of values for $x$. 
Back to Supervised Learning

**Supervised learning:**

- Make predictions, classify data, perform regression
- Dataset: \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\)
- 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
  \[
  \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)
  \]
- 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.
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

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

Weights/Parameters
Training the Model

- Parametrize it by weights/parameters
- Minimize the loss

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

Best parameters = best function \( f \)

Linear model class \( f \)

Square loss

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

\[
= \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + x_i^T \theta - y_i)^2
\]
How Do We Minimize?

• Need to solve something that looks like $\min_{\theta} g(\theta)$
  – Optimization problem; many algorithms

• **Gradient descent:**
  – start at some $\theta^{(0)}$
  – repeat till convergence:

$$\theta^{(j+1)} = \theta^{(j)} - \gamma \nabla g(\theta^{(j)})$$

Gradient of the loss, evaluated at current sol.
How Do We Minimize?

• Gradient descent
  – You’ll implement this in HW5

• Popular in practice: 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? “Generalizes!”
- To evaluate this, use a test set. Do not train on it!

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \quad (x_{n+1}, y_{n+1}), \ldots, (x_{n+p}, y_{n+p})$$

Training Data

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 2.2:** 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.
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.
Linear Regression

Simplest type of regression problem.

- **Inputs**: \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\)
  - \(x\)'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 \(\theta_0, \theta\)
Linear Regression Setup

• Goal: figure out how to minimize square loss

• Train set \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\)
  
  – Model \(f(x) = \theta_0 + x^T \theta\), wrap intercept: \(f(x) = x^T \theta\)
  
  – Take train data and make it a matrix 
    \[ X \in \mathbb{R}^{n \times d} \]
  
  – Then, square loss is 
    \[
    \frac{1}{n} \sum_{i=1}^{n} (x_i^T \theta - y_i)^2 = \frac{1}{n} \|X \theta - y\|^2
    \]
Finding The Optimal Parameters

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

• Could optimize it with GD, SGD, etc...
• Explicit formula for the minimum
  $\hat{\theta} = (X^TX)^{-1}X^Ty$

Hat: indicates an estimate

• Why use GD/SGD when we have explicit formula?
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”) on training set

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

- Small residuals: fit **training** set well
- May want to use a **test** set to check
Linear Regression $\rightarrow$ 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 $\rightarrow$ $p$ close to 1
- If really negative exp is huge $\rightarrow$ $p$ close to 0

“Logistic Regression”