CS 165B: Machine Learning
Unsupervised learning (Part 3) and Advanced Topic

CS 165B
Nov 28, 2023

Instructor: Prof. Yu-Xiang Wang
Announcement

• Poll results for “Advanced Topic” today
  • 50% Deep Learning  (Take a Deep Learning course 190I)
  • 40% Generalization  (Take a Statistical Machine Learning course!)
  • 30% Reinforcement Learning
  • 20% Online learning
  • 20% “Ask me anything”

• UCSB Course Evaluation form
  • 2% participation bonus for every one if we get above 75%
  • Great job so far. We went from 3% to 27% now. Keep it up!
Last time

• Clustering:
  • “Loss function” for k-means
  • Lloyd’s algorithm --- Alternating Minimization.
  • Generative model for clustering: Gaussian Mixture Model
  • How to go nonlinear?

• Dimension-reduction
  • Why dimension reduction?
  • Finding the best low-dimensional subspace to approximate the data
  • Principal Component Analysis
Recap: Clustering

• Clustering aims at finding a partition of the data that makes sense.

• Our solutions
Recap: Clustering

- Clustering aims at finding a partition of the data that makes sense.

- Our solutions

K-Means
Recap: Clustering

• Clustering aims at finding a partition of the data that makes sense.

• Our solutions

K-Means  Gaussian Mixture Models
Recap: Clustering

- Clustering aims at finding a partition of the data that makes sense.

- Our solutions

  - K-Means
  - Gaussian Mixture Models
  - Kernel k-means?
Recap: Dimension Reduction

- Finding a low-dimensional subspace to represent the data by learning an “orthogonal bases” of the subspace.

\[
\min_{\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \min_{\alpha_1, \ldots, \alpha_k} \| \mathbf{x}_i - \sum_{j=1}^{k} \mathbf{v}_j \alpha_j \|^2
\]
Recap: Optimization problems to solve

• K-means:

$$\min_{\mu_1, \ldots, \mu_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \min_{j \in [k]} \|x_i - \mu_j\|^2$$

• Principal Component Analysis:

$$\min_{\mu, v_1, \ldots, v_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \min_{\alpha_1, \ldots, \alpha_k} \left\|x_i - \left(\mu + \sum_{j=1}^{k} v_j \alpha_j\right)\right\|^2$$

What are the similarities and differences of these two problems?
Magical algorithm from Numerical Linear Algebra --- Singular Value Decomposition

\[ M = U \Sigma V^T \]

- **Left Singular Vectors**: \( U \)
- **Singular Values**: \( \Sigma \)
- **Right Singular Vectors**: \( V \)

\[ U^T U = I_m \]
\[ V^T V = I_n \]

\[ u_i^T v_j = 0 \text{ if } i \neq j \]
\[ u_i^T v_i = 1 \]

(Picture from Wikipedia)
Recap: Principal Component Analysis

**Input:** data matrix $X$, number of principal components $k$

1. Calculate the mean $\hat{\mu}$ of the rows of $X$ by $\hat{\mu} = \frac{1}{n} \sum_{i} x_i$

2. Run SVD on the de-meaned data matrix: $X - 1 \cdot \hat{\mu}^T = U \Sigma V^T$

(Use [scipy.linalg.svd](https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.svd.html))

**Output:** the mean $\hat{\mu}$, and **the first $k$ columns of $V$** as the principal components.

**Note:**
* SVD solves the best-low-rank approximation of the sample covariance matrix. PCA hopes to explain the data by focusing on the most prominent directions of the variance in the data distribution.
** Improved computational efficiency when $k$ is small: replace full-SVD with skinny SVD that takes $k$ as an input.  
  (Use [scipy.sparse.linalg.svds](https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.svds.html))
Principal Component Analysis

1. How to perform dimension reduction for new data points after finding the principal components?

\[
\min_{\mu, v_1, \ldots, v_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \min_{\alpha_1, \ldots, \alpha_k} \| x_i - (\mu + \sum_{j=1}^{k} v_j \alpha_j) \|^2
\]

2. How to reconstruct the original data using the coefficients?

\[
\begin{align*}
\text{Input: } x. & \quad \min_{\alpha} \| x - (\mu + V\alpha) \|^2 \\
& \quad 2V^T(V\alpha - c_{x-u}) = 0 \\
& \quad V^TV\alpha = V^T(c_{x-u}) \\
& \quad \alpha = (V^TV)^{-1}V^T(c_{x-u}) = V^T(c_{x-u})
\end{align*}
\]
Example of PCA on Gaussian data
Applications of PCA to image compression

You will learn how to do this in HW4 Q3!
Going non-linear dimension reduction

- Mixture of linear subspaces:
  - Subspace clustering
  - Mixture of probabilistic PCAs
  - A combination clustering and dim-reduction

- Kernel PCA
  - Run PCA on the kernel matrix instead of the covariance matrix

- Laplacian Eigenmaps (also the related Isomap)
  - First construct a nearest neighbor graph
  - Then run SVD on the Laplacian matrix of the graph

- Neural approaches:
  - Autoencoders / variational autoencoders
  - Transformers (for data-reconstruction)
Kernel PCA on the two-moon example
Increasing the number of principal components in kernel PCA improves the reconstruction
Back to the clustering example. Let’s do dimension-reduction on kernel PCA before running k-means... Now it works!
Advanced Topics

• Generalization (Remaining time Today)
  • Master theorem of Generalization via Rademacher Complexity.
  • Examples
  • Learning Goal: understand the nature of the generalization theory.

• Deep Learning (Next Tuesday before the review)
  • A zoo of model architectures
  • MLP, ConvNet, Recurrent Neural Networks, Graph Neural Networks, Transformers
  • Learning Goal: When to use what?
Recap: Loss, Empirical Risk, and Risk

• Loss function

\[ \ell(h, (x, y)) \]

• Empirical Risk function

\[ \hat{R}(h, \text{Data}) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, (x_i, y_i)) \]

• (Population) Risk function

\[ R(h, \mathcal{D}) = \mathbb{E}_{\mathcal{D}}[\ell(h, (x_i, y_i))] \]
Recap: Four classifiers of interest

• Bayes Optimal classifier:  \( h_{\text{Bayes}} = \arg \min_h R(h) \)

• Optimal classifier  
  • within hypothesis class  
    \( h^* = \arg \min_{h \in \mathcal{H}} R(h) \)

• ERM Classifier:  
  \( \hat{h}_{\text{ERM}} = \arg \min_{h \in \mathcal{H}} \hat{R}(h) \)

• My classifier:  
  \( \hat{h} = \text{My\_Learning\_Algorithm}(\text{Data}) \)
Recap: Risk Decomposition

\[ \mathbb{E}[R(\hat{h})] - R(h_{\text{Bayes}}) \leq \mathbb{E}[\hat{R}(\hat{h}) - \hat{R}(h_{\text{ERM}})] + R(h^*) - R(h_{\text{Bayes}}) + \mathbb{E}[R(\hat{h}) - \hat{R}(\hat{h})] \]

- **Optimization Error**: How close am I from minimizing the empirical risk?
- **Approximation Error**: How much worse the best “representable” classifier is from the best classifier out there.
- **Generalization Error**: How different the empirical risk of my classifier is from its population risk?

Make sure you understand what each kind of error means!
Idea to tackle the generalization error \( \ell(h_1, h_2, h_j) = ? \)

Because \( h \in \mathcal{H} \)

\[
R(\hat{h}) - \hat{R}(\hat{h}) \leq \left| R(\hat{h}) - \hat{R}(\hat{h}) \right| \leq \max_{h \in \mathcal{H}} \left| R(h) - \hat{R}(h) \right|
\]

- Recall that

\[
\left| \mathbb{E}_{z \sim \mathcal{D}} [\ell(h, z)] - \frac{1}{n} \sum_{i=1}^{n} \ell(h, z_i) \right| \leq \varepsilon
\]

- It suffices to show that empirical mean is close to population mean for all hypothesis in the family. This is known as “Uniform Convergence”
There are many way to prove “Uniform Convergence”. My favorite one is what is known as a Rademacher Complexity

- **Empirical Rademacher Complexity** of a hypothesis class

  $\hat{\text{Rad}}(\mathcal{H}) = \mathbb{E} \left[ \max_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right]$

  where $\mathbb{P}[\sigma_i = -1] = \mathbb{P}[\sigma_i = 1] = 0.5$ i.i.d.

- **Rademacher Complexity** of a hypothesis class

  $\text{Rad}(\mathcal{H}) = \mathbb{E}_{x_1, \ldots, x_n \sim D} \mathbb{E} \left[ \max_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right]$
Rademacher complexity measures the extent to which the hypothesis class is capable of fitting purely random labels!

\[
\widehat{\text{Rad}}(\mathcal{H}) = \mathbb{E} \left[ \max_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right]
\]

• Simplest hypothesis class: Singleton hypothesis class?

\[
\widehat{\text{Rad}}(\mathcal{H}) = 0
\]

• Most complex hypothesis class: the space of all Boolean functions!

\[
\widehat{\text{Rad}}(\mathcal{H}) = \mathbb{E} \left[ \frac{1}{n} \sum_{i=1}^{n} G_i \cdot G_i' \right] = 1
\]

\[h : \mathcal{X} \rightarrow \{\pm 1\}, \quad h(x_i) = G_i\]
Rademacher complexity measures the extent to which the hypothesis class is capable of fitting purely random labels!

\[
\hat{\text{Rad}}(\mathcal{H}) = \mathbb{E} \left[ \max_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right]
\]

- Example 1: Finite hypothesis class.
  \[
  \hat{\text{Rad}}(\mathcal{H}) \leq \sqrt{\frac{2 \log |\mathcal{H}|}{n}}
  \]

- Example 2: Linear classifier
  \[
  \hat{\text{Rad}}(\mathcal{H}) \leq \sqrt{\frac{2(d + 1) \log n}{n}}
  \]

- Example 3: Data space is discrete, e.g., \(\{0,1\}^d\)?
  \[
  |\mathcal{H}| \leq 2^{2d}
  \]
Small Rademacher Complexity implies generalization

**Theorem:** With probability at least $1 - \delta$, for $\forall h \in \mathcal{H}$,

$$|\hat{R}(h) - R(h)| \leq \text{Rad}(\mathcal{H}) + \sqrt{\frac{\log(2/\delta)}{n}}$$

Example 1: Decision stump on data from $\{0,1\}^d$

$$|1-1| = d \cdot 2 + 2 = 2(d+1)$$

Example 2: Linear classifier

$$\sqrt{\frac{2(d+1)}{n}} + \sqrt{\frac{\log \frac{2}{\delta}}{n}}$$
Shattering Number and VC-Dimension

• **Shattering**: We say $\mathcal{H}$ shatters a set of points $S \subset \mathcal{X}$ if it can realizes any labeling of the points in $S$.

• **VC-dimension**: $\text{VCdim}(\mathcal{H})$ is the largest number of points in $S \subset \mathcal{X}$ such that $\mathcal{H}$ shatters $S$.

• **Example 1**: $\mathcal{X} = \mathbb{R}$ and $h \in \mathcal{H}$ satisfies
  - $h(x) = 1$ if $x \geq \tau$ and $h(x) = -1$ otherwise.
  - VC-dimension = ?

```
VCdim(\{1\}) = 1
```
Exercise: What is the VC-Dimension of?

• **Example 2**: $\mathcal{X} = \mathbb{R}$ and $h \in \mathcal{H}$ satisfies
  - $h(x) = 1$ if $x \in [a, b]$ and $h(x) = -1$ otherwise.
  - $\text{VC-Dim}(\mathcal{H}) = ?$

• **Example 3**: $\mathcal{X} = \mathbb{R}^2$, $\mathcal{H}$ contains all linear classifiers
  - $h(x) = \text{sign}(w^T x + b)$
  - $\text{VC-Dim}(\mathcal{H}) = ?$
Examples of VC dimensions in terms of the shape of the “discriminative function”

<table>
<thead>
<tr>
<th>Class $\mathcal{A}$</th>
<th>VC dimension $V_\mathcal{A}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{A} = {A_1, \ldots, A_N}$</td>
<td>$\leq \log_2 N$</td>
</tr>
<tr>
<td>Intervals $[a, b]$ on the real line</td>
<td>2</td>
</tr>
<tr>
<td>Discs in $\mathbb{R}^2$</td>
<td>3</td>
</tr>
<tr>
<td>Closed balls in $\mathbb{R}^d$</td>
<td>$\leq d + 2$</td>
</tr>
<tr>
<td>Rectangles in $\mathbb{R}^d$</td>
<td>$2d$</td>
</tr>
<tr>
<td>Half-spaces in $\mathbb{R}^d$</td>
<td>$d + 1$</td>
</tr>
<tr>
<td>Convex polygons in $\mathbb{R}^2$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>Convex polygons with $d$ vertices</td>
<td>$2d + 1$</td>
</tr>
</tbody>
</table>

Table 1: The VC dimension of some classes $\mathcal{A}$. 
Bounding Rademacher Complexity with VC-dimension

\[
\widehat{\text{Rad}}(\mathcal{H}) \leq \sqrt{\frac{2 \text{VCdim}(\mathcal{H}) \log n}{n}}
\]

**Example:** Linear classifier

\[
\widehat{\text{Rad}}(\mathcal{H}) \leq \sqrt{\frac{2(d + 1) \log n}{n}}
\]

**Example:** ReLU Neural Networks with width W and depth L

VC-Dim (NN) = WL log W

Summary: Generalization Theory

• The generalization error depends on the “Complexity” of the hypothesis class.

• The more the hypothesis class can overfit to random labels, the more it is prone to overfitting
  • Rademacher Complexity
  • VC Theory
On Deep Learning

• We have covered a fair bit of it in Lecture 15 and HW4 Q1.

• Some useful facts about deep learning to know:
  1. It is a drop-in replacement for the linear “score function” with a nonlinear one.

  2. It is a feature expansion technique but with learned features.

  3. It comes You should think of it as a “Programming Language” for constructing hypothesis classes that are suitable for your problem.
You can use neural network for all kinds of ML problems that we learned: classification, regression, clustering, dimension reduction etc..

• Neural networks provide a learnable function approximation

• Different kinds of NNs architecture (like LEGO blocks) are designed to address different challenges in different kind of problems:
  • Feedforward neural network
  • Recurrent neural network
  • Boltzmann machine
  • Convolutional neural network
  • Graph Neural Networks
  • Transformers
  • etc., etc.
Learning ≈ Configuring the Learnable Function so it behaves as instructed.
Learning \approx Configuring the Learnable Function so it behaves as instructed.

- Speech Recognition
  \[ f(\text{speech}) = \]

- Handwritten recognition
  \[ f(\text{handwritten}) = \]

- Weather forecast
  \[ f(\text{weather}) = \]

- Play video games
  \[ f(\text{game}) = \]
Learning ≈ Configuring the Learnable Function so it behaves as instructed.

• Speech Recognition

\[ f(\text{声波图}) = “你好” \]

• Handwritten recognition

\[ f(2) = “2” \]

• Weather forecast

\[ f(\text{周四}) = “周六” \]

• Play video games

\[ f(\text{游戏界面}) = “move left” \]
Generally speaking, you need to make decisions about

• Which loss function to use
  • Regression, classification, clustering, dimension reduction, but also ranking, recommendation, and others...

• What type of neural network to use
  • Images
  • Text
  • Graphs (node and edges)
  • Time series
  • Decide on the hyperparameters: Depth, Width, number of hidden units, etc...

• How to train the neural network?
  • Initialization of weights: iid random? Recale or not?
  • Optimizer to use: SGD, ADAM, etc...

• How to collect, pre-process the data...
Final review next Tuesday!

• More on deep learning... before the final review.

• In-class final next Thursday.
  • 75 minutes
  • Two cheat-sheets (you can bring the one from midterm)
This course is only an basic introduction to machine learning... Where to go from here?

- **190I Deep Learning**: Convolutional Neural Networks, Recurrent Neural Networks, Graph Neural Networks, Transformers, Optimization for Deep Learning

- **165A Artificial Intelligence**: Problem solving, Search, Planning, Logic and how learning can be used in each one of these tasks

- **292X Special topic courses**
  - Convex Optimization
  - Reinforcement Learning
  - Statistical Machine Learning
  - (Graduate-Level) Introduction to Machine Learning

- More advanced textbooks:
  - [Probabilistic Machine Learning](#) by Kevin Murphy.
  - [ML Theory book](#) by Tong Zhang.
  - [Patterns, Predictions, and Actions](#) by Recht and Hardt (2023).