

RECITATION 7

LEARNING THEORY AND GENERATIVE MODELS

10-301/10-601: INTRODUCTION TO MACHINE LEARNING

3/23/2022

1 Learning Theory

Some Important Definitions and Theorems

1. Basic notations:

- **True function** (expert/oracle) $c^* : X \rightarrow Y$ (unknown)
- **Hypothesis space** \mathcal{H} and **hypothesis** $h \in \mathcal{H} : X \rightarrow Y$
- Probability Distribution p^* (unknown)
- Training Dataset $S = \{x^{(1)}, \dots, x^{(N)}\}$

2. **True Error (expected risk)**

$$R(h) = P_{x \sim p^*(x)}(c^*(x) \neq h(x))$$

3. **Train Error (empirical risk)**

$$\begin{aligned}\hat{R}(h) &= P_{x \sim S}(c^*(x) \neq h(x)) \\ &= \frac{1}{N} \sum_{i=1}^N \mathbb{1}(c^*(x^{(i)}) \neq h(x^{(i)})) \\ &= \frac{1}{N} \sum_{i=1}^N \mathbb{1}(y^{(i)} \neq h(x^{(i)}))\end{aligned}$$

4. **PAC criterion** is that we produce a high accuracy hypothesis with high probability. More formally,

$$P(\forall h \in \mathcal{H}, |R(h) - \hat{R}(h)| \leq \epsilon) \geq 1 - \delta$$

5. A hypothesis $h \in \mathcal{H}$ is **consistent** with training data S if $\hat{R}(h) = 0$ (zero training error/correctly classify)

6. **Sample Complexity** is the minimum number of training examples N such that PAC criterion is satisfied for a given ϵ (arbitrarily small error) and δ (with high probability)

7. Sample Complexity for 4 Cases: See Figure 1. Note that

	Realizable	Agnostic
Finite $ \mathcal{H} $	Thm. 1 $N \geq \frac{1}{\epsilon} [\log(\mathcal{H}) + \log(\frac{1}{\delta})]$ labeled examples are sufficient so that with probability $(1 - \delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.	Thm. 2 $N \geq \frac{1}{2\epsilon^2} [\log(\mathcal{H}) + \log(\frac{2}{\delta})]$ labeled examples are sufficient so that with probability $(1 - \delta)$ for all $h \in \mathcal{H}$ we have that $ R(h) - \hat{R}(h) \leq \epsilon$.
Infinite $ \mathcal{H} $	Thm. 3 $N = O(\frac{1}{\epsilon} [\text{VC}(\mathcal{H}) \log(\frac{1}{\epsilon}) + \log(\frac{1}{\delta})])$ labeled examples are sufficient so that with probability $(1 - \delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.	Thm. 4 $N = O(\frac{1}{\epsilon^2} [\text{VC}(\mathcal{H}) + \log(\frac{1}{\delta})])$ labeled examples are sufficient so that with probability $(1 - \delta)$ for all $h \in \mathcal{H}$ we have that $ R(h) - \hat{R}(h) \leq \epsilon$.

12

Figure 1

- **Realizable** means $c^* \in \mathcal{H}$
 - **Agnostic** means c^* may or may not be in \mathcal{H}
8. **VC dimension** of a hypothesis space \mathcal{H} is the maximum number of points such that there exists at least one arrangement of these points and a hypothesis $h \in \mathcal{H}$ that is consistent with any labelling of this arrangement of points.
 9. If $\text{VC}(\mathcal{H}) = n$, then for all placements of $n + 1$ points, there exists no hypothesis $h \in \mathcal{H}$ that can shatter any of it.

Questions

1. For the following examples, write whether or not there exists a dataset with the given properties that can be shattered by a linear classifier.
 - 2 points in 1D
 - 3 points in 1D
 - 3 points in 2D
 - 4 points in 2D

How many points can a linear boundary (with bias) classify exactly for d-Dimensions?

- Yes
- No
- Yes
- No

$d + 1$

2. Consider a rectangle classifier (i.e. the classifier is uniquely defined 3 points $x_1, x_2, x_3 \in \mathbb{R}^2$ that specify 3 out of the four corners), where all points within the rectangle must equal 1 and all points outside must equal -1

(a) Which of the configurations of 4 points in figure 2 can a rectangle shatter?

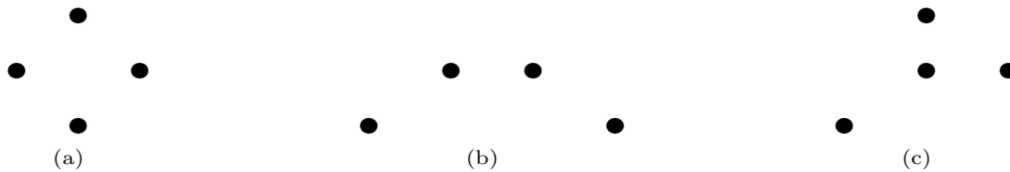


Figure 2

(a), (b), since the rectangle can be scaled and rotated it can always perfectly classify the points. (c) is not perfectly classifiable in the case that all the exterior points are positive and the interior point is negative.

(b) What about the configurations of 5 points in figure 3?

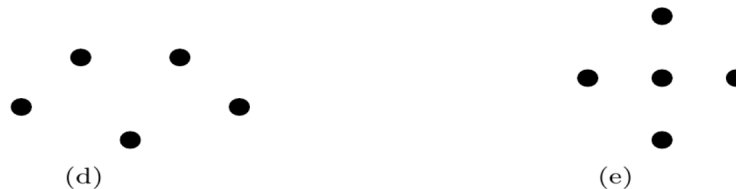


Figure 3

None of the above. For (d), consider (from left to right) the labeling 1, 1 -1, -1, 1. For (e), same issue as (c).

3. Let x_1, x_2, \dots, x_n be n random variables that represent binary literals ($x \in \{0, 1\}^n$). Let the hypothesis class H_n denote the conjunctions of no more than n literals in which each variable occurs at most once. Assume that $c^* \in H_n$.

Example: For $n = 4$, $(x_1 \wedge x_2 \wedge x_4), (x_1 \wedge \neg x_3) \in H_4$

Find the minimum number of examples required to learn $h \in H_{10}$ which guarantees at least 99% accuracy with at least 98% confidence.

$$|H_n| = 3^n$$

$$|H_{10}| = 3^{10}, \epsilon = 0.01, \delta = 0.02$$

$$N(H_{10}, \epsilon, \delta) \geq \lceil \frac{1}{\epsilon} [\ln |H_{10}| + \ln \frac{1}{\delta}] \rceil = \lceil 1489.81 \rceil = 1490$$

2 MLE/MAP

As a reminder, in MLE, we have

$$\begin{aligned}\hat{\theta}_{MLE} &= \arg \max_{\theta} p(\mathcal{D}|\theta) \\ &= \arg \min_{\theta} -\log(p(\mathcal{D}|\theta))\end{aligned}$$

For MAP, we have

$$\begin{aligned}\hat{\theta}_{MAP} &= \arg \max_{\theta} p(\theta|\mathcal{D}) \\ &= \arg \max_{\theta} \frac{p(\mathcal{D}|\theta)p(\theta)}{\text{Normalizing Constant}} \\ &= \arg \max_{\theta} p(\mathcal{D}|\theta)p(\theta) \\ &= \arg \min_{\theta} -\log(p(\mathcal{D}|\theta)p(\theta))\end{aligned}$$

Imagine you are a data scientist working for an advertising company. The advertising company has recently run an ad and they want you to estimate its performance. The ad was shown to N people. $Y^{(i)} = 1$ if person i clicked on the ad and 0 otherwise. Thus $\sum_i^N y^{(i)} = k$ people decided to click on the ad. Assume that the probability that the i -th person clicks on the ad is θ and the probability that the i -th person does not click on the ad is $1 - \theta$.

1. Note

$$p(\mathcal{D}|\theta) = p((Y^{(1)}, Y^{(2)}, \dots, Y^{(k)})|\theta) = \theta^k(1 - \theta)^{N-k}$$

Calculate $\hat{\theta}_{MLE}$.

$$\begin{aligned}\hat{\theta}_{MLE} &= \arg \min_{\theta} -\log(p(\mathcal{D}|\theta)) \\ &= \arg \min_{\theta} -\log(\theta^k(1 - \theta)^{N-k}) \\ &= \arg \min_{\theta} -k * \log(\theta) - (N - k) \log(1 - \theta)\end{aligned}$$

Setting the derivative equal to zero yields

$$\begin{aligned}0 &= \frac{-k}{\theta} + \frac{(N - k)}{1 - \theta} \\ \implies \theta_{MLE} &= \frac{k}{N}\end{aligned}$$

2. Suppose $N = 100$ and $k = 10$. Calculate $\hat{\theta}_{MLE}$.

$$\hat{\theta}_{MLE} = \frac{k}{N} = 0.10$$

3. Your coworker tells you that $\theta \sim \text{Beta}(\alpha, \beta)$. That is:

$$p(\theta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)}$$

Recall from lecture that $\hat{\theta}_{MAP}$ for a Bernoulli random variable with a Beta prior is given by:

$$\hat{\theta}_{MAP} = \frac{k + \alpha - 1}{N + \alpha + \beta - 2}$$

Suppose $N = 100$ and $k = 10$. Furthermore, you believe that in general people click on ads about 6 percent of the time, so you, somewhat naively, decide to set $\alpha = 6 + 1 = 7$, and $\beta = 100 - 6 + 1 = 95$. Calculate $\hat{\theta}_{MAP}$.

$$\hat{\theta}_{MAP} = \frac{k + \alpha - 1}{N + \alpha + \beta - 2} = \frac{10 + 7 - 1}{100 + 102 - 2} = \frac{16}{200} = 0.08$$

4. How do $\hat{\theta}_{MLE}$ and $\hat{\theta}_{MAP}$ differ? Argue which estimate you think is better.

Both estimates are reasonable given the available information. Note that $\hat{\theta}_{MAP}$ has lower variance than $\hat{\theta}_{MLE}$, but $\hat{\theta}_{MAP}$ is more biased. If you believe that this advertisement is similar to advertisements with a 6 percent click rate, then $\hat{\theta}_{MAP}$ may be a superior estimate, but if the circumstances under which the advertisement was shown were different from the usual, then $\hat{\theta}_{MLE}$ might be a better choice.

5. Suppose you are an avid BTS stan who monitors the social media accounts of each of the members. Suppose you wish to find the probability that a BTS member will post at any time of day. Over three days you look on Instagram and find the following number of new posts:

$$x = [3, 4, 1].$$

A fellow stan tells you that this comes from a Poisson distribution. That is:

$$p(x|\theta) = \frac{e^{-\theta}\theta^x}{x!}$$

Also, you are told that

$$\theta \sim \text{Gamma}(2, 2)$$

That is:

$$p(\theta) = \frac{1}{\Gamma(\alpha)\beta^\alpha} \theta^{(\alpha-1)} e^{-\frac{\theta}{\beta}}, \quad \theta > 0$$

Calculate $\hat{\theta}_{MAP}$.

(Writeup from https://en.wikipedia.org/wiki/Conjugate_prior#Practical_example)

Note:

$$\begin{aligned}
 p(\mathcal{D}|\theta) &= \frac{e^{-\theta}\theta^3}{3!} \frac{e^{-\theta}\theta^4}{4!} \frac{e^{-\theta}\theta^1}{1!} \\
 \hat{\theta}_{MAP} &= \arg \min_{\theta} -\log(p(\mathcal{D}|\theta)p(\theta)) \\
 &= \arg \min_{\theta} -\log\left(\frac{e^{-\theta}\theta^3}{3!} \frac{e^{-\theta}\theta^4}{4!} \frac{e^{-\theta}\theta^1}{1!} \frac{1}{\Gamma(2)2^2} \theta^{(2-1)} e^{-\frac{\theta}{2}}\right) \\
 &= \arg \min_{\theta} -\log e^{-3\theta} \theta^8 \theta^{(2-1)} e^{-\frac{\theta}{2}} \\
 &= \arg \min_{\theta} -\log e^{-3\theta - \frac{\theta}{2}} \theta^{8+2-1} \\
 &= \arg \min_{\theta} -\left(-3\theta - \frac{\theta}{2}\right) + (8+2-1)\log(\theta)
 \end{aligned}$$

Setting the derivative equal to zero yields

$$\begin{aligned}
 0 &= -3 - \frac{1}{2} + \frac{(7+2)}{\theta} \\
 \implies \theta_{MAP} &= \frac{7+2}{3+\frac{1}{2}} = 2.57142857143
 \end{aligned}$$

3 Convolutional Neural Networks

3.1 Dance Dance Convolution

Consider the following 4 x 4 image and 2x2 filter below.

1	3	-2	4
0	8	6	5
2	1	-9	0
4	-1	3	7

1	2
-2	-1

1. Assume that there is no padding and stride = 1. What are the dimensions of the output, and what is the value in the bottom right corner of the output image? **output is 3x3, and the bottom right value is $-9 + 0 - 6 - 7 = -22$.**
2. Now assume that we having padding = 1. Given that, what are the new dimensions of the output, and the new value in the bottom right corner? **output is now 5x5, and bottom right value is $7 + 0 + 0 + 0 = 7$.**

3. Suppose we have below 3D input I and filter F. What is the dimensions of the output, assume padding=0 and stride=1, and what is the value in the bottom right corner of the output image? **output is 1x2x3, and the bottom right value is $-2 + 6 + 3 + 1 - 1 = 7$.**

$$I[0, :, :] = \begin{pmatrix} 4 & 2 & 1 & 3 \\ 6 & 1 & -2 & -3 \\ -1 & 3 & 4 & 0 \end{pmatrix} \quad I[1, :, :] = \begin{pmatrix} -2 & 1 & 4 & 5 \\ 0 & 6 & 3 & 6 \\ 4 & -1 & 1 & 1 \end{pmatrix}$$

$$F[0, :, :] = \begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix} \quad F[1, :, :] = \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}$$

3.2 Concepts

1. What are filters?

- Filters (also called kernels) are feature extractors in the form of a small matrix used in convolutional neural layers. They usually have a width, height, depth, stride, padding, channels (output) associated with them.

2. What are convolutions?

- We sweep the filter around the input tensor and take matrix dot products based on factors such as filter size, stride, padding. The matrix dot products form a new tensor, which is the output of a convolutional layer.

3. What are the shapes of the input and output tensors¹ of a CNN layer?

- **W/o batches:** Input - $(C \times H \times W)$, Output - $(C' \times H' \times W')$
- **W/batches:** Input - $(B \times C \times H \times W)$, Output - $(B \times C' \times H' \times W')$
- Channels, Height, and Width may change when going from input to output, but number of batches won't

4. What are some benefits of CNNs over fully connected (also called dense) layers?

- Good for image-related machine learning (learns the kernels that do feature engineering)
- Pseudo translational invariance (local spatial coherence from convolutions and pooling)
- Parameter efficient

¹tensors are the generalization of matrices to multi-dimensions

3.3 Parameters

Suppose that we want to classify images that belong to one of ten possible classes (i.e. [cat, dog, bird, turtle, ..., horse]). The images come in RGB format (one channel for each color), and are downsampled to dimension 128x128.

Figure 4 illustrates one such image from the MS-COCO dataset².



Figure 4: Image of a horse from the MS-COCO dataset, downsampled to 128x128

We construct a Convolutional Neural Network that has the following structure: the input is first max-pooled with a 2x2 filter with stride 2 and 3 output channels. The results are then sent to a convolutional layer that uses a 17x17 filter of stride 1 and 12 output channels. Those values are then passed through a max-pool with a 3x3 filter with stride 3 and also 12 output channels. The result is then flattened and passed through a fully connected layer (ReLU activation) with 128 hidden units followed by a fully connected layer (softmax activation) with 10 hidden units. We say that the final 10 hidden units thus represent the categorical probability for each of the ten classes. With enough labeled data, we can simply use some optimizer like SGD to train this model through backpropagation.

Note: By default, please assume we have bias terms in all neural network layers unless explicitly stated otherwise.

1. **Draw a diagram that illustrates the channels and dimensions of the tensors before and after every neural net operation.**

Step 1:

[3@128x128] (input) -> [3@?x?] (maxpool) -> [12@?x?] (conv)
 -> [12@?x?] (maxpool) -> [?] (flatten) -> [128] (fc) -> [128] (ReLU)
 -> [10] (fc) -> [10] (softmax)

Step 2:

[3@128x128] (input) -> [3@64x64] (maxpool) -> [12@48x48] (conv)

²<https://cocodataset.org/>

-> [12@16x16] (maxpool) -> [3072] (flatten) -> [128] (fc) -> [128] (ReLU)
 -> [10] (fc) -> [10] (softmax)

Step 3:

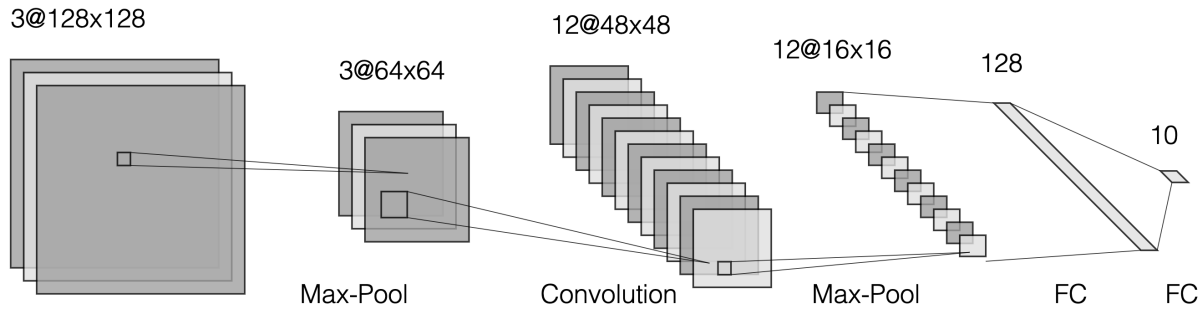


Figure 5: Full CNN structure, illustrated

2. How many parameters are in this network for the convolutional components?

$$N_{\text{conv}} = (3 \times 12 \times 17 \times 17 + 12) \\ = 10416$$

3. How many parameters are in this network for the fully connected (also called dense) components?

$$N_{\text{fc}} = (3072 \times 128 + 128) + (128 \times 10 + 10) \\ = 393344 + 1290 \\ = 394634$$

4. From these parameter calculations, what can you say about convolutional layers and fully connected layers in terms of parameter efficiency³? Why do you think this is the case?

$$\begin{aligned}N_{\text{total}} &= 10416 + 394634 \\ &= 405050\end{aligned}$$

$$N_{\text{conv}}/N_{\text{total}} = 2.57\%$$

$$N_{\text{fc}}/N_{\text{total}} = 97.43\%$$

Convolutional layers are much more parameter efficient, mainly because we are reusing the convolutional filter repeatedly for each convolutional layer (we only need to train one kernel per channel per layer). In comparison, the fully connected layer requires all nodes between two layers to be fully connected.

³the ratio between the number of parameters from some layer type and the total number of parameters.

3.4 Links

Visualization of convolutional filter sweep steps https://github.com/vdumoulin/conv_arithmetic

Visualization of convolutional filter smooth sweep with outputs <https://www.youtube.com/watch?v=f0t-0CG79-U>

Visualization of neural network layer outputs <http://cs231n.stanford.edu/>

The architecture used there is (conv → relu → conv → relu → pool) x3 → fc → softmax