

Homework 2 -Statistical Learning Theory - Due Jan 30 - Spring 2025

Essay Question: *VC theory has had mixed results in successfully explaining the generalization ability of various neural network model sets..* Write a one paragraph essay that starts with this sentence and goes on to identify and describe the successes and failures of VC theory in predicting the generalization ability of various neural network models. Be sure to provide examples supporting your assertions and conclude your essay with a single sentence summarizing the main findings of your essay.

Problem 1: Consider a model set \mathcal{H} with a finite VC dimension of 3 and let $h^* \in \mathcal{H}$ be a model that minimizes the empirical risk on a training dataset, \mathcal{D} , with 10000 samples.

1. Determine the generalization error, ϵ , for this model, h^* , with a 90% confidence level.
2. Let \mathcal{D}' be a dataset with N' samples that is statistically independent of the original dataset, \mathcal{D} , used to obtain the model h^* . Determine the size of the dataset, \mathcal{D}' , needed to obtain the generalization error that is 10 times smaller than the generalization error you found in the preceding problem with 90% confidence.
3. How many times larger or smaller is N' than the size of the original dataset \mathcal{D} ? Comment on the significance of this difference?

Solution: The first question is a simple application of the VC generalization error bound

$$\Omega(N, d_{\text{vc}}, \delta) = \sqrt{\frac{8}{N} \log \left(\frac{4((2N)^{d_{\text{vc}}} + 1)}{\delta} \right)} = \sqrt{\frac{8}{10^4} \log \left(\frac{4((2(10^4))^3 + 1)}{0.1} \right)} = \boxed{0.1635}$$

The second question takes the optimal model and applies it to a dataset with N samples. Since there is only one model in the model set now, we can use the earlier Hoeffding bound on the sample complexity for finite model sets, with $M = 1$.

$$N_{\epsilon, \delta} \geq \frac{1}{2\epsilon^2} \log \frac{2}{\delta} = \frac{1}{2(.1635.10)^2} \log \frac{2}{.1} = \boxed{5600}$$

So you only need $100 \times \frac{5600}{100,000} = 5.6\%$ of the samples needed in \mathcal{D} . In general, dataset \mathcal{D} is used to train a model, but dataset \mathcal{D}' is used to estimate the trained model's true risk. This shows that at least for this situation, the testing dataset only has to be about 5% of the size of the training dataset to obtain an estimate of the true risk that is 10 times better than that predicted by the training loss.

Problem 2: A coin is weighted so its probability of landing heads is 20%, independently of the other flips. Suppose a coin is flipped 15 times. Determine the probability that 75% of the 15 tosses are heads. Then use the Markov, Chebyshev, and Chernoff inequalities to bound the probability that 75% of the 15 tosses are heads. Compute the ratio of each bound over the actual probability of the event and comment on how much better the Chernoff bound is versus the other two.

Hint: To compute determine the tightest Chernoff bound you will need to know that the moment generating function of a Binomial random variable $X \sim \text{Binom}(n, p)$ is $\mathbb{E}[e^{\lambda X}] = ((1-p) + pe^{\lambda})^n$ where $\lambda > 0$.

Solution: Let X be the random variable representing the number of heads in $n = 15$ tosses for a coin whose probability of landing heads, $p = 0.2$. X will have a Binomial distribution, $\text{Binom}(n, p)$ where

$$\Pr(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

X takes discrete non-negative values so we want to compute $\Pr(X = \alpha n) = \Pr(X = \lfloor \alpha n \rfloor)$. In our case $\alpha = 0.75$ and $n = 15$ so we want

$$\Pr(X \geq \lceil 0.75 \times 15 \rceil) = \Pr(X \geq 12) = \sum_{k=12}^{15} \binom{15}{k} (.2)^k (.8)^{15-k} = \boxed{1.0113 \times 10^{-6}}$$

We now compute the Markov bound. Since X has a Binomial distribution, we know $\mathbb{E}[X] = np$. Since X is a non-negative random variable, we can use the bound directly to deduce

$$\Pr(X \geq \alpha n) = \Pr(X \geq \lceil \alpha n \rceil) \leq \frac{\mathbb{E}[X]}{\lceil \alpha n \rceil} = \frac{pn}{\lceil \alpha n \rceil} = \boxed{0.25}$$

We now compute the Chebyshev bound. Because X has a Binomial distribution we know its variance is $\sigma^2 = np(1 - p)$. Note that

$$\begin{aligned} \Pr(X \geq \lceil \alpha n \rceil) &= \Pr(X - np \geq \lceil \alpha n \rceil - np) \leq \Pr(|X - np| \geq \lceil \alpha n \rceil - np) \\ &\stackrel{\text{cheby}}{\leq} \frac{\text{var}(X)}{(\lceil \alpha n \rceil - np)^2} = \frac{np(1 - p)}{(\lceil \alpha n \rceil - np)^2} = \frac{15(.2)(.8)}{(12 - 3)^2} = \boxed{.0296} \end{aligned}$$

We now compute the Chernoff bound. To do this we need the moment generating function for a Binomial random variable, $X \sim \text{Binom}(n, p)$. The hint gives us this

$$\mathbb{E}(e^{\lambda X}) = ((1 - p) + pe^{\lambda})^n$$

So using the Chernoff bound we get

$$\Pr(X \geq \lceil \alpha n \rceil) \leq \min_{\lambda > 0} \frac{\mathbb{E}[e^{\lambda X}]}{e^{\lambda \lceil \alpha n \rceil}} = \min_{\lambda > 0} \frac{((1 - p) + pe^{\lambda})^n}{e^{\lambda \lceil \alpha n \rceil}} = \min_{\lambda > 0} \frac{(.8 + .2e^{\lambda})^{15}}{e^{\lambda 12}} = \boxed{3.8147 \times 10^{-6}}$$

The boxed quantity was obtained by first plotting $\frac{(0.8+0.2e^{\lambda})^{15}}{e^{\lambda 12}}$ and finding the value of λ that minimizes this quantity. That plot is shown in Fig. 1. Taking the ratio of the bounds to the actual probability we get the markov bound being 247,218 times larger than actual probability, the Chebyshev bound being 29,300 times larger than actual probability, and the Chernoff bound being 4 times larger than actual probability. So the Chernoff bound is at least 4 orders of magnitude better than the other bounds.

Notebook Assignment 1 (TensorFlow): This assignment uses the [fashion-mnist dataset](#). Fashion-MNIST is a dataset of Zalando's article images; consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28 by 28 grayscale image associated with a label from 10 classes.

```

prob = 0
p = 1/5;n=15;alf = 3/4;
for k=ceil(n*alf):n;
    prob = prob + nchoosek(n,k)*p^k*(1-p)^(n-k);
end
prob

%markov bound
markov_bound = p*n/(ceil(alf*n))

%chebyshev bound
cheby_bound = n*p*(1-p)/(ceil(alf*n)-n*p)^2

lambda = 0:.01:8;
chernx = (((1-p)+p*exp(lambda)).^n)./exp(lambda*ceil(alf*n))
chernoff = ((.8+.2*exp(lambda)).^15)./exp(12*lambda);
plot(lambda,chernoff,linewidth=2)
grid on
xlabel('lambda')
ylabel('Pr(X>12)')

axis([0 8 0 1.e-3])
figure(1)
chernoff_bound = min(chernoff)

%ratios
int64([markov_bound cheby_bound chernoff_bound]/prob)

```

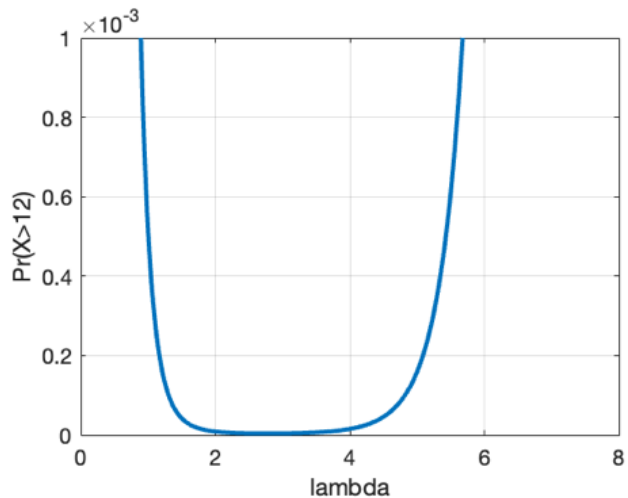


Figure 1: Plot of Chernoff bound

1. Load the Fashion-MNIST dataset's training and testing data (samples and targets). Retype the input samples as float32 and check the shape of the arrays for the training/testing samples and targets.
2. Find an image for each type of garment and display that image with the title of that garment over the top of the image. Arrange your 10 images in a 2 (rows) by 5 (columns) grid.
3. Instantiate a tensorflow model with two dense layers. The input layer has shape (28*28,). The next dense layer has 128 nodes with an relu activation. The last (output) layer has 10 nodes with a softmax activation. Compe the model with an adam optimizer, sparse categorical crossentropy loss function, using "accuracy" as the performance metric. Build the model and print the model summary.
4. Reshape the training and test samples to have shape (60000,28*28). Rescale the sample to [0,1] and make sure they are typed as float32. Split the training data into a p-training and validation set using a 2/3 split. Create a callback to save the "best model" with the lowest validation loss. Then Train the model on the p-training data using the validation data to create the validation curves. Train your model for 30 epochs assuming a batch size of 512 and return the history object.
5. Create a function `plot_training_curves` that takes the `history` object and plots the ptraining/validation loss in one figure and the ptraining/validation accuracy in the other figure. Then evaluate the best model's accuracy on the test data.

Solution: See jupyter notebook for solution.