DDA2020: Homework III

April 10, 2022

Homework due: **11:59pm, April 24, 2022**.

1 Written Problems

1. Given the following loss function, please plot the computational graph , and derive the update procedure of parameters using back-propagation algorithm,

$$\begin{aligned}\mathcal{L}(\mathbf{W},\mathbf{b}) =& CE\Big(y, \sigma\big(\max(0, W_3 \tanh(W_1\boldsymbol{x} + b_1) + b_2) + \\ & \tanh(W_4 \max(0, W_2\boldsymbol{x} + b_3) + b_4)\big)\Big) + \lambda \sum_{i=1}^{4} \|W_i\|_2^2,\end{aligned} \tag{1}$$

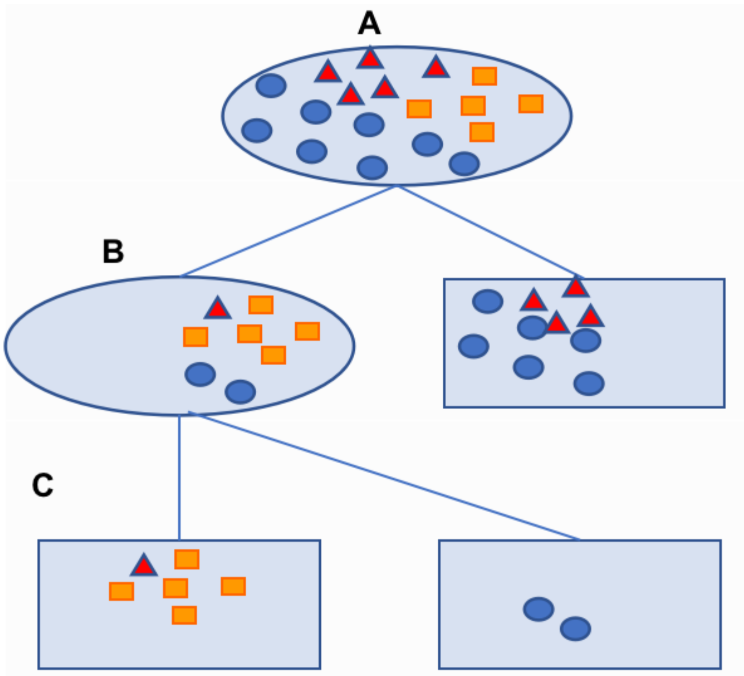
   where $\mathbf{W} = \{W_1, W_2, W_3, W_4\}, \mathbf{b} = \{b_1, b_2, b_3, b_4\}$ denote the parameters; $\boldsymbol{x} \in \mathbb{R}^d$ indicates the input features; $y \in \mathbb{R}$ is the ground-truth label. (2 points)

2. The input shape is $63 \times 63 \times 3$, and the CNN model has 4 layers, *i.e.*, $Conv_1 + Maxpool_1 + Conv_2 + Maxpool_2$.

   - $Conv_1$: 10 $5 \times 5 \times 3$ filters, stride=2, padding=2
   - $Maxpool_1$: 2×2 filter, stride=3, padding=0
   - $Conv_2$: 20 $4 \times 4 \times 10$ filters, stride=2, padding=1
   - $Maxpool_2$: 2×2 filter, stride=2, padding=1

   Please compute the shape of activation map of each layer, the number of parameters (hint: don't forget the bias parameter for each convolution filter), the computational cost of the forward pass. (1 point)

3. Compute the Gini index, the entropy and the classification error for each node of the tree in the figure below. (1 point)



- Entropy: $\phi(p, 1-p) = -p\log_2 p - (1-p)\log_2(1-p)$
- Gini Index: $\phi(p, 1-p) = 2p(1-p)$
- Misclassification Error: $\phi(p, 1-p) = 1 - \max(p, 1-p)$

4. (2 points) Suppose we randomly sample a training set D from some unknown distribution. For each training set D we sample, we train a regression model h_D to predict y from x (one dimensional). We repeat this process 10 times resulting in 10 trained models.
Recall that $y = t(x) + \epsilon$, where $\epsilon \in \mathcal{N}(0, \sigma^2)$. Here, we specify $\sigma^2 = 0.5$.
For a new test sample $(x, y) = (3, 7)$ sampled from the same distribution that generated the training sets, we suppose $t(x = 3) = 6.7$, and ϵ is instantiated as 0.3.
Suppose the predictions of the new test sample based on the 10 trained models are 6, 8, 9, 5, 10, 5, 4, 8, 9, 3.
(a) Based on this 10 trials, please compute the empirical mean squared error (MSE), Bias^2 and Variance on this test sample.(1 point)
(Hint:

- For a new test sample (x, y), we define its mean squared error (MSE) by different models as

$$MSE(x, y) = E_D[(h_{D_i}(x) - y)^2].$$

- It can observed that $E_{(x,y),D}[(h_{D_i}(x) - y)^2] = E_{(x,y)}[MSE(x, y)]$.

- The empirical estimation of MSE based on above 10 trained models is

$$\widehat{MSE}(x,y) = \frac{1}{10}\sum_{i=1}^{10}(h_D(x) - y)^2.$$

(b) Explain why $\widehat{MSE}(x,y) \neq \text{Bias}^2 + \text{Variance} + \sigma^2$ (1 point)

5. (2 points) Neural networks with different activation functions.
Consider a two-layer network function of the form

$$y_k(\mathbf{x},\mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right),$$

in which the hidden unit nonlinear activation function $h(\cdot)$ is given by logistic sigmoid function of the form

$$\sigma(a) = \frac{1}{1+\exp(-a)}.$$

Show that there exists an equivalent network, which computes exactly the same function, but with the hidden unit activation function given by tanh(a) where the tanh function is defined by

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}.$$

That is, if there's another two-layer network function with tanh(a) as hidden unit activation function:

$$\hat{y}_k(\mathbf{x},\hat{\mathbf{w}}) = \sigma\left(\sum_{j=1}^{M} \hat{w}_{kj}^{(2)} \tanh\left(\sum_{i=1}^{D} \hat{w}_{ji}^{(1)} x_i + \hat{w}_{j0}^{(1)}\right) + \hat{w}_{k0}^{(2)}\right),$$

then there exists linear transformation between these w and $\hat{w}$, that enable $y_k(x,w) = \hat{y}_k(x,\hat{w})$ for all x.
Hint: first find the relation between $\sigma(a)$ and $\tanh(a)$, and then show that the parameters of the two networks differ by linear transformations.

6. (*Optional*) Connection of single-layer regression network to pursuit regression model
First, consider a pursuit regression model. Assume we have an input vector X with p components, and a target Y. Let $\omega_m, m = 1, 2, \ldots, M$, be unit p-vectors of unknown parameters. The projection pursuit regression (PPR) model has the form

$$f(X) = \sum_{m=1}^{M} g_m\left(\omega_m^T X\right)$$

This is an additive model, but in the derived features $V_m = \omega_m^T X$ rather than the inputs themselves. The functions g_m are unspecified and are estimated along with the directions ω_m using some flexible smoothing

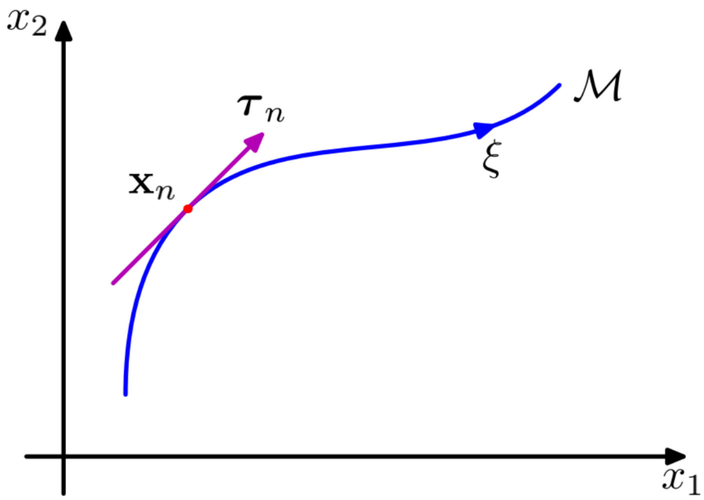


Figure 1: Illustration of a two-dimensional input space showing the effect of a continuous transformation on a particular input vector $\mathbf{x}_n$. A one dimensional transformation, parameterized by the continuous variable ξ, applied to $\mathbf{x}_n$ causes it to sweep out a one-dimensional manifold $\mathcal{M}$. Locally, the effect of the transformation can be approximated by the tangent vector τ_n.

method as follows. The function $g_m\left(\omega_m^T X\right)$ is called a ridge function in $\mathbb{R}^p$. It varies only in the direction defined by the vector ω_m. The scalar variable $V_m = \omega_m^T X$ is the projection of X onto the unit vector ω_m, and we seek ω_m so that the model fits well, hence the name "projection pursuit."

We want you to establish the exact correspondence between the projection pursuit regression model (1) and the neural network. In particular, show that the single-layer regression network is equivalent to a PPR model with $g_m\left(\omega_m^T x\right) = \beta_m \sigma\left(\alpha_{0m} + s_m\left(\omega_m^T x\right)\right)$, where ω_m is the m th unit vector. Establish a similar equivalence for a classification network. Recall the neural network for K-class classification is

$$
\begin{aligned}
Z_m &= \sigma\left(\alpha_{0m} + \alpha_m^T X\right), m = 1, \ldots, M \\
T_k &= \beta_{0k} + \beta_k^T Z, k = 1, \ldots, K \\
f_k(X) &= g_k(T), k = 1, \ldots, K
\end{aligned}
$$

where $Z = (Z_1, Z_2, \ldots, Z_M)$, and $T = (T_1, T_2, \ldots, T_K)$. See Fig1 for illustration.

7. (*Optional*) Consider a binary classification problem in which the target values are $t \in \{0, 1\}$, with a network output $y(\mathbf{x}, \mathbf{w})$ that represents $p(t = 1 \mid \mathbf{x})$, and suppose that there is a probability ϵ that the class label on a training data point has been incorrectly set. Assuming independent and

identically distributed data, write down the error function corresponding to the negative log likelihood. Verify that the cross-entropy error function

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln (1 - y_n)\}$$

(here y_n denotes $y(\mathbf{x}_n, \mathbf{w})$) is obtained when $\epsilon = 0$. Note that this error function makes the model robust to incorrectly labelled data, in contrast to the usual error function.

2 Programming

1. Decision Tree (required) (8 points)

Task description Fit (*i.e.*, regression) the real variable *Sales* in the **Carseats** dataset, using decision tree, bagging, and random forests. All these algorithms can be implemented by calling **sklearn** in Python. The loss is set as sum of squared error (SSE).

Dataset **Carseats** contains 400 data points (saved in 400 rows). For each data, the first column is the value of target variable *Sales* that we want to fit; the remaining 9 columns indicate 9 features (or attributes), as shown in Fig. 3. There is no fixed train/test splitting. In this project, you have two options: simply set the first 300 rows as the training set, and the remaining 100 rows as the testing set; randomly split the whole dataset to 300 train + 100 test, and try multiple times.

What you should do

- **Data statistics**: analyze the statistics of the target variable and each feature, and try to visualize the statistics (*e.g.*, histogram) (0.5 point)
- **Decision tree**: solve the above problem using decision tree method; report the train/test errors with respect to different maximum depths, different least node sizes; plot the learned tree (2 points)
- **Bagging of trees**: solve the above problem using the bagging method, with decision tree as the base learner; report the train/test errors with respect to different depths, different number of trees (2 points)
- **Random forests**: solve the above problem using the random forest method, with decision tree as the base learner; report the train/test errors with respect to different number of trees, different values of m (the number of candidate attributes to split in every step, see Slides 'W7-Decision Tree', Page 68) (2 points)
- Plot the curve of bias^2 with respect to different number of trees in random forests, *e.g.*, $\#tree = 10, 20, \ldots, 100$. Then, describe the relationship between bias^2 and different number of trees; repeat the procedure for variance. (1.5 points)

Note: You should use Python. Finally, you should submit one self-included code file (without external dependencies) and a technical report(PDF, with written questions)
Please name your code file as "A3_StudentMatriculationNumber.py" and report as and "A3_StudentMatriculationNumber.pdf", while "A3_MatricNumber"

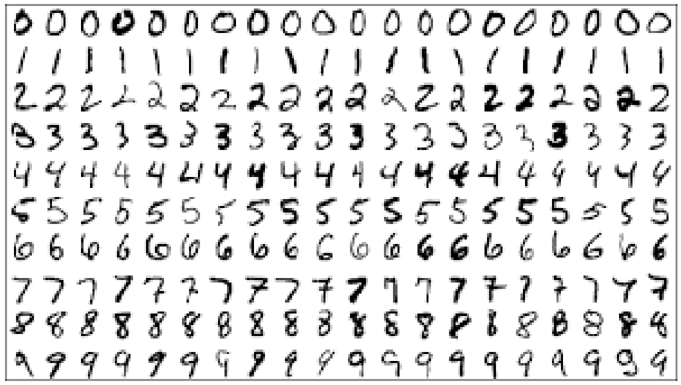
using your student matriculation number. For example, if your matriculation ID is 123456789, then you should submit "A3_123456789.py" and "A3_123456789.pdf".

The reference report is in Tutorial1. You can check in on BlackBoard. (You can submit several files in one submission. Don't submit them in different submission.)

Carseats

Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
9.5	138	73	11	276	120	Bad	42	17	Yes	Yes
11.22	111	48	16	260	83	Good	65	10	Yes	Yes
10.06	113	35	10	269	80	Medium	59	12	Yes	Yes
7.4	117	100	4	466	97	Medium	55	14	Yes	Yes
4.15	141	64	3	340	128	Bad	38	13	Yes	No
10.81	124	113	13	501	72	Bad	78	16	No	Yes
6.63	115	105	0	45	108	Medium	71	15	Yes	No
11.85	136	81	15	425	120	Good	67	10	Yes	Yes
6.54	132	110	0	108	124	Medium	76	10	No	No
4.69	132	113	0	131	124	Medium	76	17	No	Yes
9.01	121	78	9	150	100	Bad	26	10	No	Yes
11.96	117	94	4	503	94	Good	50	13	Yes	Yes
3.98	122	35	2	393	136	Medium	62	18	Yes	No
10.96	115	28	11	29	86	Good	53	18	Yes	Yes
11.17	107	117	11	148	118	Good	52	18	Yes	Yes
8.71	149	95	5	400	144	Medium	76	18	No	No
7.58	118	32	0	284	110	Good	63	13	Yes	No
12.29	147	74	13	251	131	Good	52	10	Yes	Yes
13.91	110	110	0	408	68	Good	46	17	No	Yes
8.73	129	76	16	58	121	Medium	69	12	Yes	Yes
6.41	125	90	2	367	131	Medium	35	18	Yes	Yes
12.13	134	29	12	239	109	Good	62	18	No	Yes
5.08	128	46	6	497	138	Medium	42	13	Yes	No
5.87	121	31	0	292	109	Medium	79	10	Yes	No
10.14	145	119	16	294	113	Bad	42	12	Yes	Yes

Figure 2: Some examples of **Carseats**.



2. GISETTE (Optional)

   Task description GISETTE is a handwritten digit recognition problem. The problem is to separate the highly confusible digits '4' and '9'. The data set was constructed from the MNIST data that is made available by Yann LeCun and Corinna Cortes at http://yann.lecun.com/exdb/mnist/. The dataset for this problem is large, so please budget time accordingly for this problem. You should use the Scikit Learn SVM package for this problem.

   - Standard run: Use all the 60000 training samples from the training set to train the model, and test over all test instances, using the linear kernel.
   - Kernel variations: In addition to the basic linear kernel, investigate two other standard kernels: RBF (a.k.a. Gaussian kernel; set γ = 0.001), Polynomial kernel (e.g. set degree=2,coef0=1; e.g, $(1 + x^T x)^2)$).

3. Handwritten Digit Recognition using PyTorch(Optional)

   Task description In this problem, you should develop a handwritten digit classifier from scratch. You will be utilizing the PyTorch package for this problem. You may find it helpful to review the PyTorch tutorials https://pytorch.org/tutorials/. You should download the MNIST dataset and design your own CNN using PyTorch. Then you train the network and see the performance(loss and accuracy) under different hyper-parameters. This can help you better understand the power of deep learing. You can refer to the following materials for help.

   References

- PyTorch Deep Explainer MNIST example: https://www.kaggle.com/code/ceshine/pytorch-deep-explainer-mnist-example/notebook
- Handwritten Digit Recognition Using PyTorch: https://towardsdatascience.com/handwritten-digit-mnist-pytorch-977b5338e627
- PyTorch tutorials: https://pytorch.org/tutorials/