Diep Vu

1. Classification and Cross-entropy loss

Part-1: Basic Concepts

x_n is an input data sample, and it is a vector.
y_n is the ground-truth class label of x_n.
$\hat{y}_n$ is the output "soft-label"/confidence of a logistic regression classifier given the input x_n
n is from 1 to N
the number of classes is K

(1) write down the formula of binary cross-entropy (K = 2), assuming y_n is an integer

$$L = -\frac{1}{N}\sum_{n=1}^{N}\left(y_n log(\hat{y}_n) + (1 - y_n) log(1 - \hat{y}_n)\right)$$

(2) write down the formula of cross-entropy (K ≥ 2), assuming y_n is a one-hot vector

$$L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} y_{(n,k)} log\left(\hat{y}_{(n,k)}\right)$$

(3) Assume there are only two classes: class-0, class-1, and the data point x_n is in class-1 (y_n = 1). Assume the output is $\hat{y}_n$ = 0.8 from a binary logistic regression classifier.
Compute the binary cross-entropy loss associated with the single data sample x_n
note: show the steps

$$\begin{aligned}\text{Cross\_entropy}(y_n, \hat{y}_n) &= -(\, y_n\, log(\hat{y}_n) + (1 - y_n)\, log(1 - \hat{y}_n)) \\ &= -(1 log(0.8) + (1 - 1) log(1 - 0.8)) \\ &= -(log(0.8) + 0) = 0.0969\end{aligned}$$

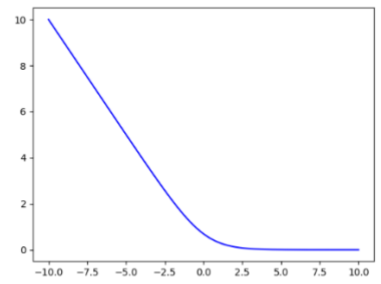
(4) Assume there are three classes: class-0, class-1 and class-2, and the data point x_n is in class-2 (y_n = 2). Assume the output is $\hat{y}_n = [0.1, 0.2, 0.7]^T$ from a multi-class logistic regression classifier
Do one-hot-encoding on y_n, and then Compute the cross-entropy loss associated with the single data sample x_n
note: show the steps

$y_n = [0\ 0\ 1]^T$

$$\begin{aligned}\text{Cross\_entropy}(y_n, \hat{y}_n) &= -\,(y_{n,0}\, log(\hat{y}_{n,0}) + y_{n,1}\, log(\hat{y}_{n,1}) + y_{n,2}\, log(\hat{y}_{n,2})) \\ &= -\,(0\, log(0.1) + 0\, log(0.2) + 1\, log(0.7)) \\ &= -\,(0 + 0 + log(0.7)) = 0.155\end{aligned}$$

(5) Show that the function $f(x) = -log\left({}^{1}/_{1+e^{-x}} \right)$ is convex in x. log is the natural log

Here is a plot of the function, and it seems that the function is convex.



Hint: show that $\partial^2 f/\partial x^2 \geq 0$ then it is convex. This explains why cross entropy loss is convex. ∂x^2

The concept of cross entropy is from information theory

$$f(x) = -\ln\left(\frac{1}{1+e^{-x}}\right)$$

$$= -\ln(1+e^{-x})^{-1}$$

$$= \ln(1+e^{-x})$$

$$\frac{df}{dx} = \frac{1}{1+e^{-x}} \cdot (-e^{-x}) = \frac{-e^{-x}}{1+e^{-x}} = -e^{-x}(1+e^{-x})^{-1}$$

$$\frac{d^2f}{dx^2} = -\left[e^{-x} \cdot -(1+e^{-x})^{-2} \cdot -e^{-x} + (1+e^{-x})^{-1} \cdot -e^{-x}\right]$$

$$= -\left[e^{-2x}(1+e^{-x})^{-2} - (1+e^{-x})^{-1} e^{-x}\right]$$

$$= -e^{-x}(1+e^{-x})^{-2}\left[\underbrace{e^{-x} - (1+e^{-x})}_{-1}\right]$$

$$= \frac{e^{-x}}{(1+e^{-x})^2} \geq 0$$

2. Regression

x_n is an input data sample, and it is a vector.
M is the number of elements/features in x_n.
y_n is the ground-truth
y_n is a vector that has **two elements** $[y_{n,1}, y_{n,2}]$. For example, $y_{n,1}$ is income, and $y_{n,2}$ is age, given input image x_n $\hat{y}_n$ is the output of a regressor (e.g., linear regressor) given the input x_n
n is from 1 to N

(1) write down the formula of MSE loss

$$MSE = \frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2$$

(2) write down the formula of MAE loss

$$MAE = \frac{1}{K}\sum_{k=1}^{K}|y_k - \hat{y}_k|$$

(3) write down the formula of MAPE loss

$$MAPE = \frac{1}{K}\sum_{k=1}^{K}\left|\frac{y_k - \hat{y}_k}{y_k}\right| \times 100\%$$

3. Decision Tree

A decision tree is a partition of the input space.
Every leaf node of the tree corresponds to a region of the final partition of the input space.

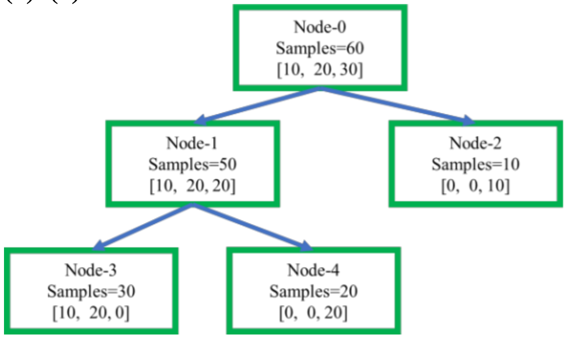
(1) The output of a decision tree for regression looks like a staircase. Why?

The output of a decision tree for regression looks like a staircase because the tree partitions the input space into regions based on the values of features. At each split, a new threshold is created for the feature, and the decision tree produces the same output within each region, which makes the structure look like a staircase.

(2) Is it a good strategy to build a deep tree such that every leaf-node of the tree is a pure node?

Constructing a deep tree is not always a good idea. While it may lead to a high accuracy on the training data, it can result in overfitting, where the model performs poorly on new, unseen data.

(3)~(6) are related to the tree below



(3) What is the total number of training samples according to the above tree?
Samples = 60

(4) What is the max-depth of the tree?
Max-depth of the tree is 2 (Node-0 → Node-1 → Node-4)

(5) How many 'pure' nodes (entropy =0) does this tree have? What is the entropy on Node-0?

There 2 pure nodes (entropy = 0), which are Node-2 and Node-4

Entropy: $H = -\sum_{k=1}^{K} \hat{p}_{(j,k)} log\hat{p}_{(j,k)}$

Entropy(Node-0) = $-\left((10/60)\, log_2(10/60) + (20/60)\, log_2(20/60) + (30/60)\, log_2(30/60)\right)$

$= -1.46$

(6) How many leaf/terminal nodes?
There are 3 terminal nodes (Node-2, Node-3 and Node-4)

4. Bagging and Random Forest

(1) Under what condition will Bagging work?

Bagging works best when the base learning algorithm has high variance, meaning that small changes in the training data can lead to significant changes in the resulting model. This is because averaging the outputs may reduce the variance of estimation if the outputs are independent and identically distributed random variables or weakly correlated.

(2) Random-forest uses a strategy to reduce the correlation between trees. What is the strategy?

Random forests help to reduce tree correlation by injecting more randomness into the tree-growing process. A random subset of the features is selected for each tree to split on, rather than using all the features at each split. This means that each tree in the forest is trained on a slightly different subset of the features, which helps to reduce the correlation between the trees.

5. Boosting

What is the difference between boosting (e.g., XGBoost) and bagging (e.g., Random-forest) from the perspective of variance and bias?

Bagging can help to reduce the variance of the model by averaging multiple models, while boosting can help to reduce the bias of the model by iteratively adjusting the model.

6. Stacking

(1) What is the difference between stacking and bagging?

Bagging, that often considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process.

Stacking, that often considers heterogeneous weak learners, learns them in parallel and combines them by training a meta-model to output a prediction based on the different weak models predictions

Bagging is used to reduce the variance of weak learners. Stacking is used to improve the overall accuracy of strong learners.

(2) Could it be useful to stack many polynomial models of the same degree?

Stacking many polynomial models of the same degree can have some benefits, but it's not always the best approach. This is because when you stack many polynomial models of the same degree, you can potentially capture more complex non-linear relationships that a single polynomial model might miss. However, it can lead to overfitting and can be computationally expensive and may not scale well to large datasets.

(3) Could it be useful to stack models of different types/structures?

Yes, it can be useful to stack models of different types/structures, as it allows you to leverage the strengths of different types of models and potentially improve the overall predictive performance.

7. Overfitting and Underfitting

It is easy to understand Overfitting and Underfitting, but it is hard to detect them. Consider two scenarios in a classification task:
(1) the training accuracy is 100% and the testing accuracy is 50%
(2) the training accuracy is 80% and the testing accuracy is 70%
In which scenario is overfitting likely present?

Overfitting is likely present in scenario (1) since training accuracy is high in contrast to the low testing accuracy
Consider two new scenarios in a classification task:
(1) the training accuracy is 80% and the testing accuracy is 70%
(2) the training accuracy is 50% and the testing accuracy is 50%
In which scenario is underfitting likely present?

Overfitting is likely present in scenario (2) since training accuracy and testing accuracy are both low.

Keep in mind that, in real applications, the numbers in different scenarios may be very similar.
We can always increase model complexity to avoid underfitting.
We need to find the model with the “right” complexity (i.e., the best hyper-parameters) to reduce overfitting if possible.

8. Training, Validation, and Testing for Classification and Regression

(1) What are hyper-parameters of a model? Give some examples.
A hyperparameter is a machine learning parameter whose value is chosen before a learning algorithm is trained. Examples: learning rate, number of epochs, degree (for a polynomial model), number of clusters,..

(2) Why do we need a validation set?
Validation set is useful for hyperparameter tuning, identifying overfitting, or selecting the best model out of different models since using a separate dataset for validation will allow us to estimate the performance of the model to unseen data.

(3) Why don't we just find the optimal hyper-parameters on the training set? e.g., find the model that performs the best on the training set.
This is because a model that performs the best on the training set may be overfitting and will not generalize well to new data. This can result in poor performance on the validation set or in real-world scenarios.

(4) Why don't we optimize hyper-parameters using the testing set?
Terminologies: training(train) set(dataset), testing(test) set(dataset), validation (val) set(dataset)
The testing set should be reserved for evaluating the final performance of the model. If you use the testing set to optimize hyperparameters, you risk overfitting to the testing set and missing the opportunity to test the performance of the model on unseen data.

9. SVM

(1) Why maximizing the margin in the input space will improve classifier robustness against noises?
Maximizing the margin in the input space can improve classifier robustness against noises because it helps to create a decision boundary that is less likely to be affected by small variations in the input data.

(2) Will the margin in the input space be maximized by a nonlinear SVM?
No, as the decision boundary in the case of the nonlinear SVM is not in a shape of the plane, so it is not possible to maximize the margin in the input space. Instead, nonlinear SVM uses kernel, a kind of 'similarity' measurement between two data points.

(3) Why could SVM cause “out-of-memory” error for a large dataset?
This is because when the training dataset is too large to fit in memory, the SVM algorithm may run out of memory while trying to allocate memory for storing the data and the kernel matrix.

(4) What is the purpose of using a kernel function?
The kernel function is used to map the original non-linear observations into a higher-dimensional space in which they become separable. This is also known as the "kernel trick," and it allows the SVM algorithm to efficiently learn nonlinear decision boundaries in the input data space.

10. Handle class-imbalance

We have a class-imbalanced dataset, and the task is to build a classifier on this dataset. From the perspective of PDF, there are two types/scenarios of class-imbalance (see lecture notes). Now, assume we are in scenario-1.

(1) Why do we use weighted-accuracy to measure the performance of a classifier? i.e., What is the problem of the standard accuracy?

Weighted accuracy takes into account the imbalanced nature of the dataset in scenario-1 by assigning different weights to each class proportional to its frequency in the dataset. This means that the accuracy of the minority class is given more importance and contributes more to the overall weighted accuracy. The problem with standard accuracy is that it is biased towards the majority class, and a classifier that always predicts the majority class can still achieve a high accuracy.

(2) When class-weight is not an option for a classifier, what other options do we have to handle class-imbalance?

One option to deal with class imbalance is resampling, which consists of up-sampling and down-sampling

11. Entropy

The PMF for a discrete random variable X is $[p_1, p_2, p_3, \dots p_K]$ and $\sum_k p_k = 1$
Write down the entropy and show that:

Entropy: H = $-\sum_k p_k \, log(p_K)$

(1) entropy is non-negative

$\sum_k p_k = 1$, using Jensen's inequality, which states that for any convex function f(x):

$f(\sum_k t_k x_k) \leq \sum_k t_k f(x_k)$, where t_k are non-negative weights such that $\sum_k t_k = 1$.

H $-\sum_k p_k \, log(p_K) \geq -\,log(\sum_k p_k \, p_K)$ since log(x) is a convex function and p_k are non-negative weights

H $\geq -\,log(\sum_k p_k{}^2) \geq -\,log(\sum_k p_k) \geq log(1) \geq 0$

Entropy H is non-negative

(2) entropy reaches the maximum when the PMF is a uniform distribution, i.e., $p_k = 1/K$
$p_k = 1/K$ for all k, meaning that all possible values of X are equally likely.
Entropy: H = $-\sum_k p_k \, log(p_K)$
$= -\sum_k (1/K) \, log(1/K)$
$= -\sum_k (1/K) \, log(1) + \sum_k (1/K) \, log(K)$
$= (1/K) \sum_k log(K)$
Since log(K) is a constant value that does not depend on the specific values of p_K, it remains unchanged as we vary the probabilities. Entropy reaches its maximum when the PMF is a uniform distribution.

Hint: you can use Jensen's inequality in Q.12 or Lagrange Multiplier