Homework 2

Divesh Badod

1 Chapter #1

Q1: What is representation learning? Give an example of it – where would it be applied? What are the factors of variation?

Ans:- Performance of various simple machine learning algorithms depends on the representation of given data. For example, for a given Logistic Regressor used to recommend caesarean delivery, the doctor feeds in a formalized report to this model with relevant pieces of information. These pieces of information included in the representation of the patient are known as features. The given model will learn how each of these features correlates with the various outcomes. However, it cannot influence the way these features are defined. Many tasks in Artificial Intelligence can be solved by designing the right set of features to use for that task. However, for most of these tasks it is difficult to know what features should be designed. Hence a solution to this problem is to use machine learning to discover not only the mapping of the representation to the output but the representation itself. This approach of finding the representation of the data is called representation learning. One of the important examples of representation learning is autoencoders which is the combination of the encoder function which covers the input data into a new representation and decoder function which reverts back this new representation to its original format. Autoencoders are trained to preserve as much information as possible when an input is run through the encoder and then the decoder but are also trained to make the new representation have various nice properties.

The features to be extracted for the representation takes an enormous amount of time, hence when designing an algorithm for learning features the goal is to usually separate the factors of variation that explains the observed data. Factors in this context refers to separate sources of influence.

Q2: Explain the relationship between an artificial neural network's architecture, e.g., number of layers, number of nodes etc., to its complexity.

Ans:- A feedforward neural network or multilayer perceptron is a mathematical function mapping some set of inputs to output values. This function is formed by composing many simpler functions. Each of these simpler functions can be thought of a new representation of the input. Each layer of the representation can be thought of as the state of

the computer's memory after executing another instruction in sequence. Networks with greater depth can execute more instructions in a sequence.

There are two main ways of measuring the depth or complexity of the model. The first is based on the number of sequential instructions that must be executed to evaluate the architecture. Which means the length of the longest path through a flow chart that describes how to compute each of the model's outputs given its inputs.

And another way is used by probabilistic models, regarding the depth of a model as being not the depth of the computational graph but the depth of the graph describing how concepts are related to each other. In this case the depth of the flowchart of the computations needed to compute the representation of each concept may be much deeper than the graph of the concepts themselves.

2 Chapter #2

Q1: What is Eigenvalue Decomposition? Explain/define (in words) what is: 1) a positive definite matrix, 2) a positive semi-definite matrix, 3) a negative definite matrix, and 4) a negative semi-definite matrix.

Ans:- Eigenvalue decomposition is a widely used matrix decomposition method in which we decompose a matrix into a set of eigenvectors and eigenvalues. Eigen decomposition of a matrix A is given by

$$A = V diag(\lambda) V^{-1}$$

Where V is a non-zero vector and λ is a scalar eigenvalue corresponding to this eigenvector.

A matrix whose eigenvalues are all positive is called positive definite matrix.

A matrix whose eigenvalues are all positive or zero-valued is called positive semidefinite matrix.

Likewise, a matrix whose eigenvalues are all negative than it is called negative definite matrix and if all eigenvalues are negative or zero-valued, it is negative semidefinite matrix.

Q2: Give 2 applications where we would use Singular Value Decomposition (SVD).

Ans:- The two applications of singular value decompositions are:

- Pseudoinverse: The SVD can be used for computing the pseudoinverse of a matrix
- Total Least squares minimization: This problem refers to determining the vector x which minimizes 2-norm of a vector Ax under the constraint $\| x \| = 1$. The solution turns out to be right-singular vector of A corresponding to the smallest singular value.

3 Chapter #3

Q1: What is the difference between absolute independence and conditional independence? Explain in words (referring to the equations in the book): what is expected value, variance, and covariance?

Ans:- Absolute Independence: Given two random variables x and y are independent if their probability distribution can be expressed as a product of two factors, one involving only x and one involving only y:

$$\forall x \in X, y \in Y, p(x = X, y = Y) = p(x = X)p(y = Y)$$

Conditional Independence: Given two random variables x and y are conditional independent given a random variable z if the condition probability distribution over x and y factorizes in the following way for every value of z:

$$\forall x \in X, y \in Y, z \in Z, p(x = X, y = Y|z = Z) = p(x = X|z = Z)p(y = Y|z = Z)$$

Expected Value: The expectation or expected value of some function f(x) with respect to a probability distribution P(x) is the average or mean value that f takes on when x is drawn from P. For discrete variables this can be computed with a summation:

$$E_{x\sim p}[f(x)] = \sum_x P(x)f(x)$$

While for continuous variables, it is computed with an integral

$$E_{x\sim p}[f(x)] = \int P(x)f(x)dx$$

Variance: Variance gives a measure of how much the values of a function of a random variable x vary as we sample different values of x from its probability distribution.

$$Var(f(x)) = E[(f(x) - E[f(x)])^2]$$

Covariance: Covariance gives some sense of how much two values are linearly related to each other, as well as the scale of these variables.

$$Cov(f(x), g(y)) = E[(f(x) - E[f(x)])(g(y) - E[g(y)]$$

Q2: Briefly explain the different kinds of statistical distributions (make sure to mention/refer to at least 4 different specific types of named distributional models). What is

a mixture model and why might it prove useful in modeling some types of data?

Ans:- Bernoulli Distribution: This is a distribution over a single binary random variable. It is controlled by a single parameter $\phi \in [0, 1]$, which gives the probability of the random variable being equal to 1. It has the following properties

$$P(x = 1) = \phi$$

$$P(x = 0) = 1 - \phi$$

$$P(x = x) = \phi^x(1 - \phi)^{1-x}$$

$$E[X] = \phi$$

$$Var_x(X) = \phi(1 - \phi)$$

Multinoulli Distribution: This is a distribution over a single discrete variable with k different states, where k is finite. The multinoulli distribution is parameterized by a vector $p \in [0, 1]^{k-1}$, where p_i gives the probability of the i-th state. The final k-th state's probability is given by $1-1^T p$. This type of distribution is often referred to distributions over categories of objects, so we do not usually assume the state has 1 numerical value 1. For this reason, we do not usually need to compute the expectation or variance of multinoulli random variables.

Gaussian Distribution: This is a type of continuous probability distribution for real-valued random variables, this is the most commonly used distribution over real numbers.

$$N(x : \mu, \sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} exp(-\frac{1}{2\sigma^2}(x - \mu)^2)$$

The two parameters $\mu \in R$ and $\sigma \in (0, \infty)$ control the normal distribution. The parameter μ gives the coordinate of central peak and is also the mean of the distribution $E[X] = \mu$. The standard deviation of the distribution is given by σ

Laplace Distribution: It is a type of continuous probability distribution use to place a sharp peak of probability mass at an arbitrary point μ.

$$L(x : \mu, \gamma) = \frac{1}{2\gamma} exp(-\frac{\mid x - \mu \mid}{\gamma})$$

Where μ is the location parameter $\gamma > 0$ which is sometimes referred to as the diversity and scale parameter.

Mixture Model: This is a probabilistic model for representing the presence of subpopulations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs. Formally a mixture model corresponds to the mixture distribution that represents the probability distribution of observations in the overall population. It is also common to define probability distributions by combining other simpler probability distributions. A mixture model is made up of several components of distributions. On each trail the choice of

which component distribution generates the sample is determined by sampling a component identity from a multinoulli distribution.

$$P(x) = \Sigma_i P(c = i)P(x \mid c = i)$$

Where P(c) is the multinoulli distribution over component identities. These models are used to create a richer distribution of data. Problems related to mixture models relate to deriving the properties of overall population from those of the sub-populations mixture models are used to make statistical inferences about the properties of the subpopulations given only observations on the pooled population, without sub-population identity information.

Q3: Suppose that we have three coloured boxes: r (red), b (blue), and g (green). Box r contains 3 apples, 4 oranges, and 3 limes, box b contains 1 apple, 1 orange, and 0 limes, and box g contains 3 apples, 3 oranges, and 4 limes. If a box is chosen at random with probabilities p(r) = 0:2, p(b) = 0:2, and p(g) = 0:6, and a piece of fruit is removed from the box (with equal probability of selecting any of the items in the box), then what is the probability of selecting an apple? If we observe that the selected fruit is in fact an orange, what is the probability that it came from the green box?

Ans:Baye's Theorem:

$$p(apple) = p(apple \mid r)p(r) + p(apple \mid b)p(b) + p(apple \mid g)p(g)$$

$$= 0.3 \times 0.2 + 0.5 \times 0.2 + 0.3 \times 0.6$$

$$= 0.34$$

$$P(o) = p(o)p(r) + p(o)p(b) + p(o)p(g)$$

$$= 0.4 \times 0.2 + 0.5 \times 0.2 + 0.3 \times 0.6$$

$$= 0.36$$

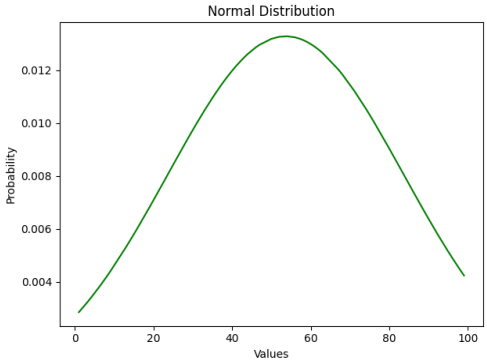
$$P(g \mid o) = p(o) * p(g)/p(o)$$

$$= \frac{0.3 \times 0.6}{0.36}$$

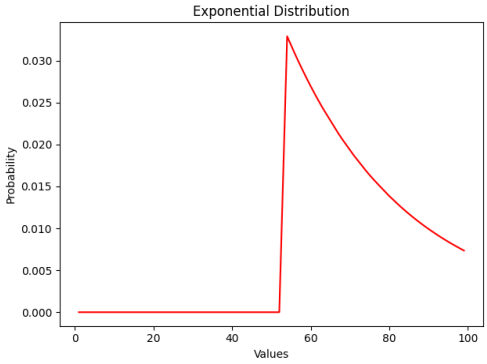
$$= 0.5$$

Q4: Generate a set of random points and plot graphs for: 1) the Gaussian distribution, 2) the exponential distribution, 3) the Laplace distribution, and 4) the Dirac distribution. Note that you will need to write code in order to solve this problem, e.g., Python.

Ans:- For seed = 3

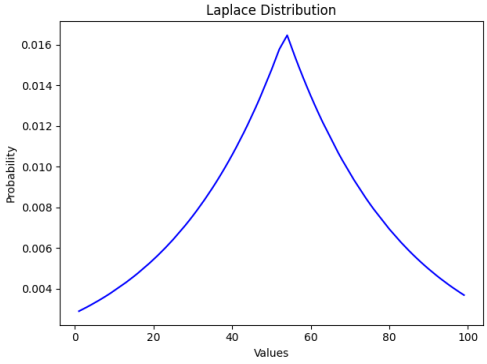


(a) Normal Distribution

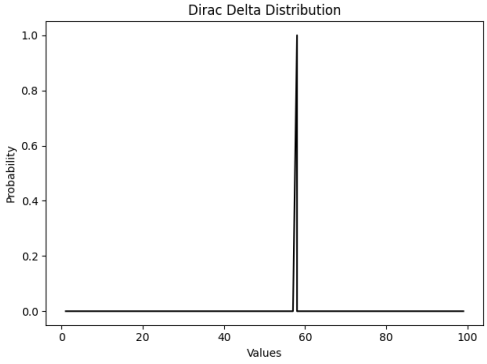


(b) Exponential Distribution

Figure 1



(a) Laplace Distribution



(b) Dirac Delta Distribution

Figure 2

Q5: What is the Kullback-Leibler (KL) Divergence? (describe it in words, referring to the equation/formula in the book). Give an example of when KL Divergence would be particularly useful.

Ans:- If we have two separate probability distributions P(x) and Q(x) over the same random variable x, we can measure how different these two distributions are using the Kullback-Leibler(KL) divergence:

$$D_{KL}(P \parallel Q) = E_{x\sim p}[logP(x) - logQ(x)]$$

The KL divergence has many useful properties most notably that it is a non-negative. If KL divergence is 0 then P and Q are the same distribution is case of discrete variables or equal in the case of continuous variables. Cross-entropy is closely related to KL divergence. Minimizing the cross-entropy with respect to Q is equivalent to minimizing the KL divergence, because Q does not participate in the omitted term.

Q6: Write the probability distribution for each of the following two graphs in the diagram below, i.e., graph a and graph b.

Ans:- Probability distribution for Graph a

$$P(a, b, c, d, e) = p(a)p(b|a)p(c|a)p(d|b, c)p(e|b)$$

Probability distribution for Graph b

$$P(a, b, c, d, e, f) = p(a)p(d|a)p(b|d, e)p(e)p(c|e)p(f|b)$$

4 Chapter #5

Q1: What is the no free lunch theorem (NFLT)? Why is it important for statistical learning in general and how would it affect your choice of a statistical model to apply for any particular problem?

Ans:- Learning theory claims that machine learning algorithm can generalize well from a finite training set of examples, which seems to contradict some basic principles of logic. Machine learning avoids to logically infer a rule which describes every member of a set given that one must have information about every member of that set, in part it avoids this problem by offering only probabilistic rules, rather than the entirely certain rules used in logical reasoning, it promises to find rules that are probably correct about most members of the set they concern. Unfortunately, this does not resolve the entire problem, hence the no free lunch theorem is an important concept. The no free lunch theorem for machine learning states that, averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points. In other words, in some sense, no machine learning algorithm is universally any better than any other.

This holds only when we average over all the possible data generating distributions. If

we make assumptions about the kinds of probability distributions we encounter in real-world applications, then we can design learning algorithms that perform well on these distributions. This means that the goal of the machine learning research is not to seek universal learning algorithm or absolute best learning algorithm. Instead the goal is to understand what kinds of distributions are relevant to the real-world that an AI agent experiences and what kinds of machine learning algorithm performs well on data drawn from the kinds of data generating distributions we care about. In this way the NFLT affects our choice of a statistical model to apply for a particular problem.

Q2: When is it is appropriate to use an L1 (Laplacian) regularization term? When is it appropriate to use an L2 (Gaussian) regularization term)? What effect does L1 and L2 regularization have on model weights?

Ans:- L1 regularization also called Lasso regularization tends to shrink coefficients to zero whereas L2 or Ridge tends to shrink coefficients evenly. Hence L1 is used for feature selection when we have huge number of features, as we can drop any variables associated with coefficients that go to zero. L2 on the other hand is useful when you have collinear/co-dependent features. Co-dependency tends to increase coefficient variance, making coefficients unstable, which hurts model generality. L2 reduces the variance of these estimates which counteracts the effect of co-dependency.

L1 and L2 are regularization techniques which are use in cases of high variance, more generally after applying regularization methods to a model we are essentially penalizing the model by adding a penalty called a regularizer to the cost function. These are also called weight decays L1 regularization is done as followed:

$$J(\theta) = MSE_{error} + \lambda|w|^1$$

L2 regularization is done as followed:

$$J(\theta) = MSE_{error} + \lambda|w|^T|w|$$

Where λ is a value that controls the strength of our preference for smaller weights. When $\lambda = 0$ we impose no preference in a choice and larger values of λ forces the weights to become smaller. Minimizing the cost function results in a choice of weights that make a trade-off between fitting the training data and being small.

Q3: Describe the steps to be taken in each case when the bias is high and when the variance is high. How do we balance the trade-off in each case?

Ans:- In supervised learning, underfitting happens when a model unable to capture the underlying pattern of the data. These models usually have high bias. It happens when we have very less amount of data to build an accurate model, when we try to build a linear model with a nonlinear data or when we have a hypothesis function which is too simple or using very few features. Hence to tackle this problem we have to create add more features to the hypothesis function. If new features are not available, we can create new features by combining two or more existing features or by taking a square, cube

and so on of the existing feature.

Overfitting happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy dataset and the models generate too complex hypothesis function with too many features. These models have high variance. These models are very complex like Decision trees which are prone to overfitting. Hence steps taken to reduce these features are to reduce the unnecessary features from the function through feature selection which is done by L1 regularization, or keep all the features in the function but reduce the magnitude of the higher order features and variable dependencies using L2 regularization.

In cases such as above we need a model which has a good balance between bias and variance which is also known as Bias-Variance Tradeoff. The most common way to negotiate this trade-off is to use cross-validation. Alternatively it can be done by minimizing the total mean squared error which is given as

$$Err(x) = Bias^2 + Variance$$

Minimizing this equation gives us a tradeoff between minimizing Bias and minimizing the variance.

Q4: When would you use the categorical cross entropy loss and the mean squared error, respectively?

Ans:- Cross-entropy is a better measure than MSE for classification, because decision boundaries in a classification task is large in comparison with regression. MSE doesn't punish misclassifications enough but is right loss for regression, where the distance between two values can be predicted is small.

From a probabilistic point of view, the cross-entropy is the natural cost function to use if you have a sigmoid or softmax non-linearity in the output layer of your model or network and you want to maximize the likelihood of classifying the input data correctly. If instead you assume the target is continuous and normally distributed, and you maximize the likelihood of the output of the network under these assumptions, you get the MSE combined with a linear output layer.