CSCE 790: Neural Networks and Their Applications

AIISC and Dept. Computer Science and Engineering
Email: vignar@sc.edu

© Vignesh Narayanan

October 31, 2023

UNIVERSITY OF
South Carolina

Machine Learning - Basic Principles

- Deep learning - a type of ML algorithm
- ML algorithms have settings called hyper parameters, which must be determined outside the learning algorithm itself
- Most deep learning algorithms are based on an optimization algorithm called stochastic gradient descent
- We have seen - Supervised learning and (a glimpse of) reinforcement learning
- We will do unsupervised learning - SOMs and Auto-encoders

Learning Algorithms

Definition (Learning - Mitchell, 1977)

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Tasks

- Machine learning tasks are usually described in terms of how the machine learning system should process an example.
- An example is a collection of features that have been quantitatively measured from some object or event that we want the machine learning system to process.
- We typically represent an example as a vector $x \in \mathbb{R}^n$, where each entry x_i of the vector is another feature

Example Tasks - Classification

- In this type of task, the computer program is asked to specify which of k categories some input belongs to.
- To solve this task, the learning algorithm is usually asked to produce a function $f : \mathbb{R}^n \to \{1, \dots, k\}$.
- When $y = f(x)$, the model assigns an input described by vector x to a category identified by numeric code y.
- There are other variants of the classification task, for example, where f outputs a probability distribution over classes.
- An example of a classification task is object recognition, where the input is an image (usually described as a set of pixel brightness values), and the output is a numeric code identifying the object in the image.

Example Tasks - Classification with Missing Inputs

- Classification becomes more challenging if the computer program is not guaranteed that every measurement in its input vector will always be provided.
- To solve the classification task, the learning algorithm only has to define a single function mapping from a vector input to a categorical output.
- When some of the inputs may be missing,rather than providing a single classification function, the learning algorithm must learn a set of functions.
- Each function corresponds to classifying x with a different subset of its inputs missing.
- This kind of situation arises frequently in medical diagnosis, because many kinds of medical tests are expensive or invasive.
- One way to efficiently define such a large set of functions is to learn a probability distribution over all the relevant variables, then solve the classification task by marginalizing out the missing variables.

Example Tasks - Regression

- In this type of task, the computer program is asked to predict a numerical value given some input.
- To solve this task, the learning algorithm is asked to output a function $f : \mathbb{R}^n \to \mathbb{R}$.
- This type of task is similar to classification, except that the format of output is different.
- An example of a regression task is the prediction of the expected claim amount that an insured person will make (used to set insurance premiums), or the prediction of future prices of securities.
- These kinds of predictions are also used for algorithmic trading.

Example Tasks - Transcription

- In this type of task, the machine learning system is asked to observe a relatively unstructured representation of some kind of data and transcribe the information into discrete textual form.
- For example, in optical character recognition, the computer program is shown a photograph containing an image of text and is asked to return this text in the form of a sequence of characters (e.g., in ASCII or Unicode format).
- Google Street View uses deep learning to process address numbers in this way.
- Another example is speech recognition, where the computer program is provided an audio waveform and emits a sequence of characters or word ID codes describing the words that were spoken in the audio recording.
- Deep learning is a crucial component of modern speech recognition systems used at major companies, including Microsoft, IBM and Google

Example Tasks - Machine Translation

- In a machine translation task, the input already consists of a sequence of symbols in some language, and the computer program must convert this into a sequence of symbols in another language.
- This is commonly applied to natural languages, such as translating from English to French.
- Deep learning has recently begun to have an important impact on this kind of task.

Example Tasks - Structured Output

- Structured output tasks involve any task where the output is a vector (or other data structure containing multiple values) with important relationships between the different elements.
- This is a broad category and subsumes the transcription and translation tasks described above, as well as many other tasks.
- One example is parsing - mapping a natural language sentence into a tree that describes its grammatical structure by tagging nodes of the trees as being verbs, nouns, adverbs, and so on.
- Another example is pixel-wise segmentation of images, where the computer program assigns every pixel in an image to a specific category.

Example Tasks - Anomaly Detection

- In this type of task, the computer program sifts through a set of events or objects and flags some of them as being unusual or atypical.
- An example of an anomaly detection task is credit card fraud detection.
- By modeling your purchasing habits, a credit card company can detect misuse of your cards.
- If a thief steals your credit card or credit card information, the thief's purchases will often come from a different probability distribution over purchase types than your own.
- The credit card company can prevent fraud by placing a hold on an account as soon as that card has been used for an uncharacteristic purchase.

Example Tasks - Synthesis and Sampling

- In this type of task, the machine learning algorithm is asked to generate new examples that are similar to those in the training data.
- Synthesis and sampling via machine learning can be useful for media applications when generating large volumes of content by hand would be expensive, boring, or require too much time.
- For example, video games can automatically generate textures for large objects or landscapes, rather than requiring an artist to manually label each pixel.
- In some cases, we want the sampling or synthesis procedure to generate a specific kind of output given the input.
- For example, in a speech synthesis task, we provide a written sentence and ask the program to emit an audio waveform containing a spoken version of that sentence.
- This is a kind of structured output task, but with the added qualification that there is no single correct output for each input, and we explicitly desire a large amount of variation in the output, in order for the output to seem more natural and realistic.

Example Tasks - Imputation of Missing Values and Denoising

- Imputation
  - In this type of task, the machine learning algorithm is given a new example $x \in \mathbb{R}^n$, but with some entries x_i of x missing.
  - The algorithm must provide a prediction of the values of the missing entries.
  - Spatio-temporal irregularites
- Denoising
  - In this type of task, the machine learning algorithm is given as input a corrupted example $\tilde{x} \in \mathbb{R}^n$ obtained by an unknown corruption process from a clean example $x \in \mathbb{R}^n$.
  - The learner must predict the clean example x from its corrupted version $\tilde{x}$, or more generally predict the conditional probability distribution $p(x|\tilde{x})$.

Example Tasks - Density Estimation

- In the density estimation problem, the machine learning algorithm is asked to learn a function $p_{model} : \mathbb{R}^n \to \mathbb{R}$, where $p_{model}(x)$ can be interpreted as a probability density function (if x is continuous) or a probability mass function (if x is discrete) on the space that the examples were drawn from.
- To do such a task well, the algorithm needs to learn the structure of the data it has seen.
- It must know where examples cluster tightly and where they are unlikely to occur.
- Most of the tasks described above require the learning algorithm to at least implicitly capture the structure of the probability distribution.
- Density estimation enables us to explicitly capture that distribution.
- In principle, we can then perform computations on that distribution to solve the other tasks as well.

Performance Measure

- To evaluate the abilities of a machine learning algorithm, we must design a quantitative measure of its performance.
- Usually this performance measure P is specific to the task T being carried out by the system.
- Examples: Accuracy, Error rate, etc.

Performance Measure

- The choice of performance measure may seem straightforward and objective, but it is often difficult to choose a performance measure that corresponds well to the desired behavior of the system.
- In some cases, this is because it is difficult to decide what should be measured.
- For example, when performing a transcription task, should we measure the accuracy of the system at transcribing entire sequences, or should we use a more fine-grained performance measure that gives partial credit for getting some elements of the sequence correct? When performing a regression task, should we penalize the system more if it frequently makes medium-sized mistakes or if it rarely makes very large mistakes?
- These kinds of design choices depend on the application.

Performance Measure

- In other cases, we know what quantity we would ideally like to measure, but measuring it is impractical.
- For example, this arises frequently in the context of density estimation.
- Many of the best probabilistic models represent probability distributions only implicitly.
- Computing the actual probability value assigned to a specific point in space in many such models is intractable.
- In these cases, one must design an alternative criterion that still corresponds to the design objectives, or design a good approximation to the desired criterion.

Experience

- Machine learning algorithms can be broadly categorized as unsupervised or supervised by what kind of experience they are allowed to have during the learning process.
- Most of the learning algorithms in this book can be understood as being allowed to experience an entire dataset.
- A dataset is a collection of many examples and sometimes, an example is called a data point.

Example

One of the oldest datasets studied by statisticians and machine learning researchers is the Iris dataset (Fisher, 1936). It is a collection of measurements of different parts of 150 iris plants. Each individual plant corresponds to one example. The features within each example are the measurements of each part of the plant: the sepal length, sepal width, petal length and petal width. The dataset also records which species each plant belonged to. Three different species are represented in the dataset

Example - Unsupervised Learning

- Unsupervised learning algorithms experience a dataset containing many features, then learn useful properties of the structure of this dataset.
- In the context of deep learning, we usually want to learn the entire probability distribution that generated a dataset, whether explicitly, as in density estimation, or implicitly, for tasks like synthesis or denoising.
- Some other unsupervised learning algorithms perform other roles, like clustering, which consists of dividing the dataset into clusters of similar examples.

Example - Supervised Learning

- Supervised learning algorithms experience a dataset containing features, but each example is also associated with a label or target.
- For example, the Iris dataset is annotated with the species of each iris plant.
- A supervised learning algorithm can study the Iris dataset and learn to classify iris plants into three different species based on their measurements.
- Supervised learning involves observing several examples of a random vector x and an associated value or vector y, then learning to predict y from x, usually by estimating $p(y|x)$.

Supervised and Unsupervised ML - Any different?

- Many machine learning technologies can be used to perform both tasks.
- For example, the chain rule of probability states that for a vector $x \in \mathbb{R}^n$, the joint distribution can be decomposed as

$$p(x) = \Pi_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

- This decomposition means that we can solve the ostensibly unsupervised problem of modeling $p(x)$ by splitting it into n supervised learning problems.

Supervised and Unsupervised ML - Any different?

- We can solve the supervised learning problem of learning $p(y|x)$ by using traditional unsupervised learning technologies to learn the joint distribution $p(x, y)$, then inferring

$$p(y|x) = \frac{p(x, y)}{\sum_{y'} p(x|y')}$$

Other Learning Paradigms

- Semi-supervised learning - some examples include a supervision target but others do not
- Some machine learning algorithms do not just experience a fixed dataset.
- For example, reinforcement learning algorithms interact with an environment, so there is a feedback loop between the learning system and its experiences.

Example - Linear Regression

- The definition of a machine learning algorithm as an algorithm that is capable of improving a computer program's performance at some task via experience is somewhat abstract.
- To make this more concrete, we present an example of a simple machine learning algorithm: linear regression.
- We start with a definition of our task T - to predict y from x by outputting $\hat{y} = w'x$.
- Next we need a definition of our performance measure, P.
- Suppose that we have a design matrix of m example inputs that we will not use for training, only for evaluating how well the model performs.
- We also have a vector of regression targets providing the correct value of y for each of these examples.
- Because this dataset will only be used for evaluation, we call it the testset.
- We refer to the design matrix of inputs as $X(test)$ and the vector of regression targets as $y(test)$.

Example - Testing

- One way of measuring the performance of the model is to compute the mean-squared error of the model on the test set
- If $\hat{y}(test)$ gives the predictions of the model on the test set, then the mean squared error is given by

$$MSE(test) = \frac{1}{m}\sum_i (\hat{y}(test) - y(test))_i^2$$

- To make a machine learning algorithm, we need to design an algorithm that will improve the weights w in a way that reduces $MSE(test)$ when the algorithm is allowed to gain experience by observing a training set - $\{y(train), x(train)\}$

Training $\rightarrow$ Testing

- The central challenge in machine learning is that our algorithm must perform well on new, previously unseen inputs - not just those on which our model was trained.
- The ability to perform well on previously unobserved inputs is called generalization (testing error=generalization error vs training error).
- How can we affect performance on the test set when we can observe only the training set?
- The field of statistical learning theory provides some answers.
- If the training and the test set are collected arbitrarily, there is indeed little we can do.
- If we are allowed to make some assumptions about how the training and test set are collected, then we can make some progress.

Training $\rightarrow$ Testing

- The training and test data are generated by a probability distribution over datasets called the data-generating process.
- We typically make a set of assumptions known collectively as the i.i.d. assumptions.
- These assumptions are that the examples in each dataset are independent from each other, and that the training set and test set are identically distributed, drawn from the same probability distribution as each other.
- This assumption enables us to describe the data-generating process with a probability distribution over a single example.
- The same distribution is then used to generate every train example and every test example.

Training $\rightarrow$ Testing

- We call that shared underlying distribution the data-generating distribution, denoted p_{data}.
- This probabilistic framework and the i.i.d. assumptions enables us to mathematically study the relationship between training error and test error.
- One immediate connection we can observe between training error and test error is that the expected training error of a randomly selected model is equal to the expected test error of that model.
- Suppose we have a probability distribution $p(x, y)$ and we sample from it repeatedly to generate the training set and the test set.
- For some fixed value w, the expected training set error is exactly the same as the expected test set error, because both expectations are formed using the same dataset sampling process.
- The only difference between the two conditions is the name we assign to the dataset we sample.

Tangible Measures

- We do not fix the parameters ahead of time and then sample both datasets.
- We sample the training set, then use it to choose the parameters to reduce training set error, then sample the test set.
- Under this process, the expected test error is greater than or equal to the expected value of training error.
- The factors determining how well a machine learning algorithm will perform are its ability to -
  1. Make the training error small
  2. Make the gap between training and test error small

Revisiting Fitting Performance

- These two factors correspond to the two central challenges in machine learning: underfitting and overfitting.
- Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set.
- Overfitting occurs when the gap between the training error and test error is too large.
- We can control whether a model is more likely to overfit or underfit by altering its capacity.

Revisiting Fitting Performance

- Informally, a model's capacity is its ability to fit a wide variety of functions.
- Models with low capacity may struggle to fit the training set.
- Models with high capacity can overfit by memorizing properties of the training set that do not serve them well on the test set.
- One way to control the capacity of a learning algorithm is by choosing its hypothesis space, the set of functions that the learning algorithm is allowed to select as being the solution.

Revisiting Fitting Performance

- For example, the linear regression algorithm has the set of all linear functions of its input as its hypothesis space.
- We can generalize linear regression to include polynomials, rather than just linear functions, in its hypothesis space.
- Doing so increases the model's capacity.
- A polynomial of degree 1 gives us the linear regression model with which we are already familiar, with the prediction $\hat{y} = wx + b$ (scalar case)
- By introducing x^2 as another feature provided to the linear regression model, we can learn a model that is quadratic as a function of x: $\hat{y} = w_1 x + w_2 x^2 + b$
- Though this model implements a quadratic function of its input, the output is still a linear function of the parameters.

Model Capacity

- So far we have described only one way of changing a model's capacity: by changing the number of input features it has, and simultaneously adding new parameters associated with those features.
- There are in fact many ways to change a model's capacity.
- Capacity is not determined only by the choice of model.
- The model specifies which family of functions the learning algorithm can choose from when varying the parameters in order to reduce a training objective.
- This is called the representational capacity of the model.

Model Capacity

- In many cases, finding the best function within this family is a difficult optimization problem.
- In practice, the learning algorithm does not actually find the best function, but merely one that significantly reduces the training error.
- These additional limitations, such as the imperfection of the optimization algorithm, mean that the learning algorithm's effective capacity may be less than the representational capacity of the model family.

Model Capacity

- Our modern ideas about improving the generalization of machine learning models are refinements of thought dating back to philosophers at least as early as Ptolemy.
- Many early scholars invoke a principle of parsimony that is now most widely known as Occam's razor(c. 1287–1347).
- This principle states that among competing hypotheses that explain known observations equally well, we should choose the "simplest" one.
- This idea was formalized and made more precise in the twentieth century by the founders of statistical learning theory.

Vapnik-Chervonenkis Dimension

- Statistical learning theory provides various means of quantifying model capacity.
- Among these, the most well known is the Vapnik-Chervonenkis dimension, or VC dimension.
- The VC dimension is defined as being the largest possible value of m for which there exists a training set of m different x points that the classifier can label arbitrarily.
- Quantifying the capacity of the model enables statistical learning theory to make quantitative predictions.
- The most important results in statistical learning theory show that the discrepancy between training error and generalization error is bounded from above by a quantity that grows as the model capacity grows but shrinks as the number of training examples increases.

Remark

- Statistical bounds provide intellectual justification that machine learning algorithms can work, but they are rarely used in practice when working with deep learning algorithms.
- This is in part because the bounds are often quite loose and in part because it can be quite difficult to determine the capacity of deep learning algorithms.
- The problem of determining the capacity of a deep learning model is especially difficult because the effective capacity is limited by the capabilities of the optimization algorithm.
- We have little theoretical understanding of the general nonconvex optimization problems involved in deep learning.

Error vs Capacity

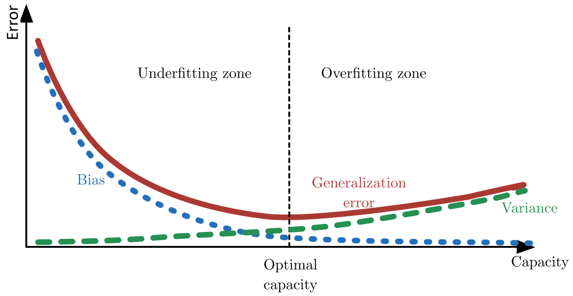

Figure: Typical relationship between capacity and error. Training and test error behave differently. At the left end of the graph, training error and generalization error are both high. This is the underfitting regime. As we increase capacity, training error decreases, but the gap between training and generalization error increases. Eventually,the size of this gap outweighs the decrease in training error, and we enter the overfitting regime, where capacity is too large, above the optimal capacity (DL, 2016).

Nonparametric Models - Examples

- Nearest neighbor regression
  - Unlike linear regression, which has a fixed-length vector of weights, the nearest neighbor regression model simply stores the x and y from the training set.
  - When asked to classify a test point x, the model looks up the nearest entry in the training set and returns the associated regression target.
- Adaptive parametrization with recursion
  - Wrapping a parametric learning algorithm inside another algorithm that increases the number of parameters as needed.
  - For example, we could imagine an outer loop of learning that changes the degree of the polynomial learned by linear regression on top of a polynomial expansion of the input.

No Free Lunch Theorem

Theorem (No free lunch theorem for ML, Wolpert, 1996)

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 ML algorithm is universally any better than any other.
- The most sophisticated algorithm we can conceive of has the same average performance (over all possible tasks) as merely predicting that every point belongs to the same class.
- The goal of machine learning research is not to seek a universal learning algorithm or the absolute best learning algorithm.
- Instead, our 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 algorithms perform well on data drawn from the kinds of data-generating distributions we care about.

Regularization

- So far, the only method of modifying a learning algorithm that we have discussed concretely is to increase or decrease the model's representational capacity by adding or removing functions from the hypothesis space of solutions the learning algorithm is able to choose from.
- The behavior of our algorithm is strongly affected not just by how large we make the set of functions allowed in its hypothesis space, but by the specific identity of those functions.
- We can also give a learning algorithm a preference for one solution over another in its hypothesis space.
- For example, we can modify the training criterion for linear regression to include weight decay.

$$J(w) = MSE(train) + \lambda w' w, \quad (\lambda = 0, \lambda >> 0?\,)$$

- Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

Hyperparameters and Validation Set

- For example, we can always fit the training set better with a higher-degree polynomial and a weight decay setting of $\lambda = 0$ than we could with a lower-degree polynomial and a positive weight decay setting.
- To solve this problem, we need a validation set of examples that the training algorithm does not observe.
- The $k-$fold cross-validation procedure - partition the dataset by splitting it into k nonoverlapping subsets.
- The test error may then be estimated by taking the average test error across k trials.
- On trial i, the i^{th} subset of the data is used as the test set, and the rest of the data is used as the training set.

Estimators, Bias and Variance

- Concepts from statistics such as parameter estimation, bias and variance are useful to formally characterize notions of generalization, underfitting and overfitting
- Point estimator: $\hat{\theta}_m = g(x_1, \ldots, x_m)$ ($\{x_i\}$ - m i.i.d. samples)
- Function estimator: $\hat{f}$ is simply the point estimator in the function space ($y = f(x) + \varepsilon$)
- Most commonly studied properties of point estimators - bias and variance

B vs V

Definition

The bias of an estimator is defined as $bias(\hat{\theta}_m) = \mathbb{E}(\theta_m) - \theta$, where the expectation is over the data (seen as samples from a random variable) and θ is the true underlying value of θ used to define the data-generating distribution

- An unbiased estimator has $bias(\hat{\theta}_m) = 0 \implies \mathbb{E}(\theta_m) = \theta$
- The variance, or the standard error (square root of variance), of an estimator provides a measure of how we would expect the estimate we compute from data to vary as we independently resample the dataset from the underlying data-generating process.
- Most common way to negotiate B vs V trade-off is to use cross-validation.
- We can also compare the mean squared error (MSE) of the estimates $MSE = \mathbb{E}[(\hat{\theta}_m - \theta)^2] = Bias(\hat{\theta}_m)^2 + var(\hat{\theta}_m)$

Derivation

- $\mathbb{E}[(\hat{\theta}_m - \theta)^2]$
- $= \mathbb{E}[(\hat{\theta}_m^2 + \theta^2 - 2\hat{\theta}_m\theta)]$
- $= \mathbb{E}(\hat{\theta}_m^2) + \mathbb{E}(\theta^2) - \mathbb{E}(2\hat{\theta}_m\theta)$
- $= var(\hat{\theta}_m) + (\mathbb{E}\hat{\theta}_m)^2 + \theta^2 - 2\mathbb{E}(\hat{\theta}_m)\theta$
- $= var(\hat{\theta}_m) + (\mathbb{E}\hat{\theta}_m - \theta)^2$
- $= Bias(\hat{\theta}_m)^2 + var(\hat{\theta}_m)$
- $var(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m - \mathbb{E}(\hat{\theta}_m))^2$

Reservoir Computing

Reservoir Observers