CMSC472: Introduction to Deep Learning

James Zhang*

February 1, 2024

These are my notes for UMD's CMSC472: Introduction to Deep Learning, which is an elective ("live-TEX"-ed). This course is taught by Assistant Professor Abhinav Shrivastava.

Contents

*Email: jzhang72@terpmail.umd.edu

§1 Introduction to Statistical Learning I

The main idea of statistical learning is to just make sense of data.

Definition 1.1. **Supervised learning**: given inputs (data-label pairs), learn a model to predict output. In mathematical terms, we want to learn a prediction function f.

Definition 1.2. **Training (or learning)**: given a training set of labeled examples $\{(x_1, y_1), \cdots, (x_N, y_N)\}$, instantiate a predictor$f$. $y = f(x) \in \mathcal{H}$, this is known as the **hypothesis space**.

Definition 1.3. **Testing (or inference)**: apply f to a new test example and return the output $f(x)$. Note that we measure correctness using loss functions.

Note 1.4. Training and testing data should be i.i.d. (independent and identically distributed) samples from the same distribution D.

§1.1 Simple Models: Classification and Regression

Definition 1.5. A **nearest neighbor classifer** returns $f(x)$ = the label of the training example nearest to x. All we need is a distance function. Note that this model requires no training.

Similarly, a **K-Nearest Neighbor** finds k nearest points and $f(x)$ = vote for class labels with labels of the k nearest points. Note that kNN is more robust to outliers.

- kNN pros: Simple to implement, decision boundaries not necessary linear, works for any number of classes, nonparametric method
- kNN cons: need good distance function, slow at test time

Definition 1.6. A **linear classifier** is a linear function that separates the classes such that $f(x) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$.

Note 1.7. Perceptrons cannot do nonlinearly separated data.

- Linear pros: low-dimensional parametric representation, easy to learn, very fast at test time
- Linear cons: works for two classes (?), how to train the linear function, data is not linearly separable

Unsuprvised Learning

Definition 1.8. Given just data as input (no labels), **unsupervised learning** models learn some sort of underlying structure.

- Clustering

- Quantization
- Dimensionality reduction, also called manifold learning.
- Density estimation, which can be used for anomoly detection. This idea is also used for training GANs.

Definition 1.9. **Weakly-Supervised Learning** is where all data needs to be labeled, but the labels are noisy (incomplete or wrong).

Definition 1.10. **Semi-supervised learning** needs nice clean labels, but there's a small portion of data that needs to labeled. If you know the underlying structure of the data, then it's fine.

Definition 1.11. **Bootstrapping** is the process of improving labelling accuracy in a semi-supervised learning sense. Bootstrapping is often used in building ML datasets.

Definition 1.12. In **active learning**, there's a human in the loop to ask questions and labels.

Definition 1.13. **Self-supervised learning / Predictive learning** uses proxy-tasks to turn an unsupervised learning problem into a supervised learning problem. It is a machine learning process where the model trains itself to learn one part of the input from another part of the input. It is also known as predictive or pretext learning. This differs from unsupervised learning because self-supervised learning is more focused on the data than the model.

Definition 1.14. **Proxy-tasks**, such as image colorization, use supervision naturally arising from data. No human provided labels. Completes a task whose labels will be useful for a supervised learning model.

Definition 1.15. **Reinforcement Learning** is the process of an agent learning in an interactive environment by sequential trial and error using feedback/rewards from its own actions and experiences.

§2 Introduction to Neural Networks I

Definition 2.1. A **neural network** is a 'differentiable' function composed out of multiple layers of computation.

Definition 2.2. A **tensor** is a d-dimensional array.

- Vector: 1-d tensor
- Matrix: 2-d tensor
- Tensor: n-d tensor

Tensors are inputs and outputs of layers.

Note 2.3. A single neuron is, very simply, one linear regression task, $\hat{y} = \mathbf{w}^T\mathbf{x} + b$. If x is an n-dimensional vector, and m is an m-dimensional vector, then w is $n \times m$ matrix and b is an m-dimensional vector, and this gives us one neuron.

Now consider three neurons stacked on top of each other. This is a fully-connected layer. Take the previous layer of output, and use it as an input to the next layer. For example,

$$\hat{y} = w_2^T(w_1^T x + b_1) + b_2 = w_2^T(w_1^T x) + w_2^T b_1 + b_2$$

You can just collapse this (distributive property).

Note 2.4. We need an f to prevent collapsing

$$\hat{y} = w_2^T f(w_1^T x + b_1) + b_2$$

and these f's give the neural networks nonlinearities. These f's are **activation functions**.

Definition 2.5. **Sigmoid** is $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ This function squashes everything between 0 and 1.

Definition 2.6. **Tanh** is $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. This function squashes everything between -0.5 and 0.5.

Definition 2.7. **Relu** is $f(x) = \max(0, x)$ which is self-explanatory.