CMSE 890-002: Mathematics of Deep Learning, MSU, Spring 2020

Lecture 03: Max. Likelihood Estimator and Functional Models

January 15, 2019

Lecturer: Matthew Hirn

1.3 Maximum likelihood estimator

In order to make our analysis more concrete and precise, let us put the general probabilistic framework of the previous section in the more specific language of discrete and continuous random variables.

Let us begin with the discrete setting. In this case $\mathcal{X}$ is a finite, but possible very large set (e.g., see the footnote for the MNIST data set in Example 1.2), and $\mathcal{Y}$ is also a finite set, for example because we are doing classification and there are a finite number of classes (again, think of MNIST, there are 10 classes). In this case, there is a probability of drawing each individual point $x \in \mathcal{X}$ into our training set $T = \{(x_i, y_i)\}_{i=1}^N$. Let us call that probability $p_X(x)$, where we use X to denote the random variable that takes values in $\mathcal{X}$ according to $p_X(x)$. Note then, the probability of drawing a point from a subset $A \subseteq \mathcal{X}$ is (equivalently, the probability that X takes a value in A):

$$\mathbb{P}_X(X \in A) = \mathbb{P}_X(A) = \sum_{x \in A} p_X(x)\,.$$

Also note, if the x_i's in the training set are sampled independently from $\mathcal{X}$ according to $p_X(x)$, then the probability of getting that particular set $\{x_i\}_{i=1}^N$ is:

$$p_X\left(\{x_i\}_{i=1}^N\right) \propto \prod_{i=1}^N p_X(x_i) = p_X(x_1) \cdot p_X(x_2) \cdots p_X(x_N)\,,$$

where the notation $\propto$ mean "proportional to." In particular, since we just care about the *set* $\{x_i\}_{i=1}^N$ and not the order in which it was drawn, the probability of drawing the set, $p_X(\{x_i\}_{i=1}^N)$, is higher than the right hand side. For example, if all the x_i's are unique (almost always the case in machine learning tasks), the correct factor is $N! = N \cdot (N-1) \cdots 2 \cdot 1$.

Recall the labels $\{y_i\}_{i=1}^N$ depend on their respective data points $\{x_i\}_{i=1}^N$. In this case, for each label $y \in \mathcal{Y}$, there is a probability of drawing y conditional on the data point being x. Let us call that probability $p_{Y|X}(y \mid x)$, where we use Y to denote the random variable that takes values in $\mathcal{Y}$ according to the conditional probabilities $p_{Y|X}(y \mid x)$. Note that if given x there is no ambiguity in the label y according to the underlying data generation process (not our model for the data!) (e.g., suppose $\mathcal{Y} = \{0, 1\}$ and for a specific x the label is always $y = 0$), then $p_{Y|X}(y \mid x)$ will be one when y is the correct label for x and zero otherwise. The

probability of drawing a label from a subset $B \subseteq \mathcal{Y}$, given that our data point is $X = x$, is:

$$\mathbb{P}_{Y|X}(B \mid X = x) = \sum_{y \in B} p_{Y|X}(y \mid x) \,.$$

The joint probability of drawing the pair (x, y) is then:

$$p_{X,Y}(x, y) = p_X(x) p_{Y|X}(y \mid x)$$

which basically says, take the probability of drawing x and multiply it by the probability of drawing y given that we drew x. It follows that the probability of drawing the training set is:

$$p_{X,Y}(T) \propto \prod_{i=1}^{N} p_{X,Y}(x_i, y_i) \,,$$

assuming again that the x_i's are drawn independently from $\mathcal{X}$ according to p_X. Note that we can extend these notions to continuous random variables as well, e.g., if $\mathcal{X} = \mathbb{R}^d$ or $\mathcal{Y} = \mathbb{R}$; see Remark 1.4 below.

Now let us describe how to model $p_{X,Y}(x, y)$ given that our only information is a single training set T. We assume $T = \{(x_i, y_i)\}_{i=1}^N$ is sampled according to the joint probability density $p_{X,Y}(x, y)$. In theory the joint probability density $p_{X,Y}(x, y)$ can be used to sample many different realizations of the data set T (e.g., in the coin tosses of Example 1.1 in which we know how the data is generated), but in practice one often does not have access to $p_{X,Y}(x, y)$ and one must make due with the given, single data set T (e.g., the MNIST data base of Example 1.2). Our goal is to find a good model for $p_{X,Y}(x, y)$ given that all we know is T. We thus consider a hypothesis space of parameterized probability distributions

$$\mathcal{P} = \{p_{X,Y}(x, y \mid \theta) : \theta \in \mathbb{R}^n\} \,,$$

where $p_{X,Y}(T \mid \theta)$ is a model that describes the probability of observing the data T given the parameters θ. If the x_i's are drawn independently from $\mathcal{X}$, then

$$p_{X,Y}(T \mid \theta) \propto \prod_{i=1}^{N} p_{X,Y}(x_i, y_i \mid \theta) \,.$$

The vector $\theta \in \mathbb{R}^n$ encodes the parameters that determine the probabilistic model $p(T \mid \theta)$. These could be the parameters of a neural network, or some other class of machine learning algorithms such as linear models or kernel methods. Since T is the only data we have, our goal is to find the model, i.e. the parameters θ, that maximizes the probability of observing T:

$$\widehat{\theta} = \arg\max_{\theta \in \mathbb{R}^n} p(T \mid \theta) \,. \tag{2}$$

Thus given T and the hypothesis space $\mathcal{P}$, our best guess for the underlying joint probability density $p_{X,Y}(x, y)$ is $p_{X,Y}(x, y \mid \widehat{\theta})$. The probability density $p_{X,Y}(x, y \mid \widehat{\theta})$ is the *maximum likelihood estimate* for the probability density $p_{X,Y}(x, y)$. We will come back to this later.

In machine learning tasks, one is often interested in the accuracy of $p_{X,Y}(x, y \mid \widehat{\theta})$ relative to $p_{X,Y}(x, y)$; that is, what is the predictive power of $p_{X,Y}(x, y \mid \widehat{\theta})$? In statistics and estimation tasks, one is concerned with the accuracy of $\widehat{\theta}$, the parameters of the model. In particular, is it significant that these particular parameters generate the optimal model from the hypothesis class $\mathcal{P}$? In either case, there are (in the current formulation) two considerations that influence the quality of the model. The first is one's ability to solve for $\widehat{\theta}$. This is not always possible, particularly in deep learning. Studying this problem amounts to studying how to optimize the parameters θ of a deep network. The second consideration is the hypothesis space $\mathcal{P}$. In particular, is it expressive enough to contain a model $p_{X,Y}(x, y \mid \theta)$ that is an accurate estimator for $p_{X,Y}(x, y)$, and furthermore can we find such a model with a finite amount of training data? In machine learning generally, this is the problem of model class selection, e.g., should we use a linear model, a quadratic model, kernel methods, or deep learning? Within deep learning, this may refer to a number of choices having to do with the architecture and design of the network, including the number of layers, the width of each layer, but also more nuanced choices such as how convolutional neural networks leverage additional structure in the data points x when x is an image.

This is a course on the mathematics of deep learning. Deep learning is, in the vast majority of cases, used for machine learning and prediction. However, in studying the mathematics of deep learning, we will attempt to understand which types of models and model classes yield good predictors $p_{X,Y}(x, y \mid \widehat{\theta})$. In certain cases this may help in understanding the role and significance of the specific $\widehat{\theta}$, although we will most likely not directly address this type of consideration; for more information in that direction, one should investigate causal inference in machine learning.

Remark 1.4. We can extend the framework at the beginning of this section to continuous random variables. In particular suppose that $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$. If X is a continuous random variable that takes values in $\mathcal{X} = \mathbb{R}^d$, then there exists a probability density function $p_X(x)$ such that

$$\mathbb{P}_X(A) = \int_A p_X(x)\, dx\,, \quad A \subseteq \mathcal{X} = \mathbb{R}^d\,.$$

Furthermore, suppose that Y conditioned on $X = x$ is a continuous random variable, which means there is a probability density function $p_{Y|X}(y \mid x)$ such that

$$\mathbb{P}_{Y|X}(B \mid X = x) = \int_B p_{Y|X}(y \mid x)\, dy\,, \quad B \subseteq \mathcal{Y} = \mathbb{R}\,.$$

The joint probability density function of X, Y is

$$p_{X,Y}(x, y) = p_X(x) p_{Y|X}(y \mid x)\,,$$

which means that the probability of A and B occurring is:

$$\mathbb{P}_{X,Y}(A, B) = \int_A \int_B p_{X,Y}(x, y)\, dy\, dx = \int_A \int_B p_X(x) p_{Y|X}(y \mid x)\, dy\, dx\,.$$

If Y is discrete, meaning without loss of generality that $\mathcal{Y} = \{1, \ldots, M\}$, we can amend the previous discussion as follows. In this case the part concerning X remains the same but for each of the M possible values of Y, we have a probability that Y takes the value conditioned on X:

$$\mathbb{P}_{Y|X}(Y = y \mid X = x) = p_{Y|X}(y \mid x)\,, \quad y \in \mathcal{Y} = \{1, \ldots, M\}\,,$$

In this case the joint probability density function is

$$p_{X,Y}(x, y) = p_X(x) p_{Y|X}(y \mid x)\,,$$

and the probability of $A \subseteq \mathcal{X} = \mathbb{R}^d$ and $B \subseteq \mathcal{Y} = \{1, \ldots, M\}$ occuring is

$$\mathbb{P}_{X,Y}(A, B) = \int_A \sum_{y \in B} p_{X,Y}(x, y)\, dx = \int_A \sum_{y \in B} p_X(x) p_{Y|X}(y \mid x)\, dx\,.$$

1.4 Supervised vs unsupervised learning

Let us clarify a bit the difference between supervised learning and unsupervised learning in this probabilistic framework that we have developed. In unsupervised learning we are only given data points $\{x_i\}_{i=1}^N \subseteq \mathcal{X}$ and we are interested in extracting patterns from the data or generating new data points; often this means our primary concern is estimating $p_X(x)$. In supervised learning, on the other hand, as we have already described we are given a training set $T = \{(x_i, y_i)\}_{i=1}^N$ consisting of data points $\{x_i\}_{i=1}^N \subset \mathcal{X}$ and labels $\{y_i\}_{i=1}^N \subset \mathcal{Y}$. Our primary concern is, given a new data point x, our ability to estimate its label $y(x)$.

Given the discussion in Sections 1.2 and 1.3, it is natural to model our candidate probability density functions $p_{X,Y}(x, y \mid \theta)$ by conditioning on x:

$$p_{X,Y}(x, y \mid \theta) = p_X(x \mid \theta) p_{Y|X}(y \mid x, \theta)\,.$$

Recall from (2) we want to maximize $p_{X,Y}(T \mid \theta)$ over all choices of $\theta \in \mathbb{R}^n$. We have:

$$\widehat{\theta} = \arg\max_{\theta \in \mathbb{R}^n} p(T \mid \theta) = \arg\max_{\theta \in \mathbb{R}^n} \prod_{i=1}^N p(x_i, y_i \mid \theta)\,,$$

since the right hand side is proportional to $p(T \mid \theta)$. Notice as well that we can take the logarithm, and still obtain the same $\widehat{\theta}$, that is:

$$\begin{aligned} \widehat{\theta} &= \arg\max_{\theta \in \mathbb{R}^n} \sum_{i=1}^N \log p(x_i, y_i \mid \theta) \\ &= \arg\max_{\theta \in \mathbb{R}^n} \left(\sum_{i=1}^N \log p_X(x_i \mid \theta) + \sum_{i=1}^N \log p_{Y|X}(y_i \mid x_i, \theta) \right)\,. \end{aligned} \tag{3}$$

Now in unsupervised learning, there are no labels, and so we only have the first summation in (3), and indeed maximizing that summation will give us the maximum likelihood estimator for $p_X(x)$. In generative modeling, we can then sample from $p_X(x \mid \widehat{\theta})$ to generate new data points. In supervised learning, remember we are primarily interested in the problem of given a new data point x, coming up with an accurate estimate for its label $y(x)$. This is encoded by the second summation, and so we often discard the first summation in this case. We will come back to this after we discuss the functional modeling perspective of supervised learning next in Section 2.

2 The supervised learning recipe

2.1 Functional models

We briefly describe the basic steps involved in supervised learning from a functional modeling perspective.

In supervised learning one is given a set of data that is partitioned into a training set $T = \{(x_i, y_i)\}_{i=1}^N$, consisting of data points $\{x_i\}_{i=1}^N \subset \mathcal{X}$ and associated labels $\{y_i\}_{i=1}^N \subset \mathcal{Y}$ that one is able to use to fit a model, and a test set T_{test} consisting of other data points and labels (x, y) not in the training set, and which are not to be used to fit the model but rather are to be used to evaluate the quality of the model fitted to the training data. Analogous to the hypothesis class $\mathcal{P}$, one defines a parameterized model class $\mathcal{F}$

$$\mathcal{F} = \{f(x;\theta) : \theta \in \mathbb{R}^n\}$$

consisting of candidate models $f(x;\theta)$ that are parameterized by $\theta \in \mathbb{R}^n$. For example, the class of linear models is given by:

$$\mathcal{F}_{\text{linear}} = \{f(x;\theta) = \langle x, \theta\rangle : \theta \in \mathbb{R}^d\}\,,$$

where $\langle x, \theta\rangle$ is the standard dot product,

$$\langle x, \theta\rangle = \sum_{i=1}^{d} x(i)\theta(i)\,.$$

One also defines a loss function $\ell(y, f(x))$ that measures the cost of a model $f(x)$ differing from a label $y = y(x)$; almost always we will take $\ell(y, f(x)) = |y - f(x)|^2$, the squared loss.

To select a model from the model class $\mathcal{F}$, we minimize the average loss over the training set:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, f(x_i;\theta))$$

$$\widehat{\theta} = \arg\min_{\theta\in\mathbb{R}^n} \mathcal{L}(\theta) = \arg\min_{\theta\in\mathbb{R}^n} \left[\frac{1}{N}\sum_{i=1}^{N} \ell(y_i, f(x_i;\theta))\right]\,.$$

We then evaluate the quality of the selected model, $f(x;\widehat{\theta})$, by evaluating it on the test set and computing the average loss over the test set:

$$\text{average test error } = \frac{1}{|T_{\text{test}}|} \sum_{(x,y)\in T_{\text{test}}} \ell(y, f(x;\widehat{\theta}))\,.$$

Similarly to the discussion in Section 1, the quality of the arrived upon model depends on two points. The first is, can one solve for the parameters $\widehat{\theta}$ or some other parameters θ that are nearly as good? Second, the choice of model class $\mathcal{F}$ is incredibly important, as it determines the set of possible models from which we will select one. There needs to be a model in $\mathcal{F}$ that simultaneously fits the training data T while at the same time does not overfit to spurious patterns in the training set so that it generalizes well to the test set. This is a complicated problem because for each new data set or task, the underlying distribution $p_{Y|X}(y \mid x)$ will change and the relationship between data point x and label $y(x)$ will thus change. We will thus want algorithms that are able to adapt to a multitude of scenarios, but which also do not need too many training points to find the (near) optimal model.

References

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations*, 2015.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

[4] Larry Greenemeier. AI versus AI: Self-taught AlphaGo Zero vanquishes its predecessor. *Scientific American*, October 18, 2017.

[5] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to Machine Learning for physicists. arXiv:1803.08823, 2018.