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

Lecture 06: Statistical Analysis and True Risk

January 24, 2019

Lecturer: Matthew Hirn

4 Basics of statistical learning theory

Figure 11 illustrates an important concept in machine learning, which is the bias-variance tradeoff. Later in Section 4.2 we will discuss this in more detail. First, though, in Section 4.1 we examine k-nearest neighbors and linear regression from a statistical point of view. We will also derive the naive Bayes classifier for classification, which will explain the purple line in Figure 11. Then in Section 4.3 we will discuss the curse of dimensionality.

4.1 Statistical view of models

We now consider k-nearest neighbors and linear regression from the perspective of statistical learning theory. Let us return to the assumptions of Section 1.3. In particular, recall that we assume we have a probability space $(\mathcal{X}, \Sigma, \mathbb{P}_X)$, $\mathcal{X} = \mathbb{R}^d$, with probability density function $p_X(x)$. We also have the label set $\mathcal{Y}$, which we will assume is $\mathcal{Y} = \mathbb{R}$. Labels are drawn from the conditional probability distribution $\mathbb{P}_{Y|X}$, which has probability density function $p_{Y|X}(y \mid x)$, and which together with $p_X(x)$ forms the joint probability density function $p_{X,Y}(x, y) = p_X(x)p_{Y|X}(y \mid x)$. We draw our training set $T = \{(x_i, y_i)\}_{i=1}^N$ from the joint probability density $p_{X,Y}(x, y)$.

Now let X be a random variable (more precisely, random vector) that takes values in $\mathcal{X}$ according to $p_X(x)$, and Y a random variable dependent upon X that takes values in $\mathbb{R}$ according to $p_{Y|X}(y \mid x)$, so that in particular the pair (X, Y) take values according to the joint probability distribution $p_{X,Y}(x, y)$. The variables X and Y, in other words, are just like the samples x_i and y_i except that we view them as random variables instead of as fixed samples.

If we had perfect knowledge of $\mathcal{X}$, $\mathcal{Y}$ and $p_{X,Y}(x, y)$; and we are using the squared loss as our measure of loss, i.e. $\ell(y, f(x)) = |y - f(x)|^2$; and we could pick any model f that we want, then we would minimize the following:

$$\begin{aligned} R_{\text{true}}(f) = \mathbb{E}_{X,Y}[(Y - f(X))^2] &= \int_{\mathcal{X}\times\mathcal{Y}} (y - f(x))^2 p_{X,Y}(x, y)\, dy\, dx \\ &= \int_{\mathcal{X}} \int_{\mathcal{Y}} (y - f(x))^2 p_{Y|X}(y \mid x)\, dy\, p_X(x)\, dx \\ &= \mathbb{E}_X \mathbb{E}_{Y|X}[(Y - f(X))^2 \mid X] \end{aligned} \tag{9}$$

The functional $R_{\text{true}}(f)$ is called the *true risk* of the model f. It measures the quality of the model f if we had an oracle that told us everything about the data generation process. In practice, of course, all we are given is the finite number of training points $T = \{(x_i, y_i)\}_{i=1}^N$. So instead of minimizing the true risk, which is almost always impossible because we simply do not have complete knowledge (in fact if we did, we would not be doing machine learning!), we instead minimize the *empirical risk*, which is the loss function we saw earlier:

$$R_{\text{emp}}(f) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i))^2 \,.$$

Note that the empirical risk is the finite sample average of $(Y - f(X))^2$. By the Law of Large Numbers, $R_{\text{emp}}(f) \to R_{\text{true}}(f)$ as $N \to \infty$ for a *fixed* f. This does not imply, though, that the minimizer of the empirical risk converges to the minimizer of the true risk as $N \to \infty$. Rather this is a topic in statistical learning theory and must be proved for each new model class.

We will study the ramifications of minimizing the empirical risk in Section 4.2. For now, let us return to the theoretical scenario now and minimize the true risk, $R_{\text{true}}(f)$. From (9) we see that we can solve for the optimal f pointwise, i.e., by finding $\widehat{f}(x)$ one x at a time. In particular, we obtain:

$$\widehat{f}(x) = \arg\min_c \mathbb{E}_{Y|X}[(Y - c)^2 \mid X = x] \,.$$

Taking the derivative with respect to c and setting it equal to zero, we obtain:

$$\widehat{f}(x) = \widehat{c} = \mathbb{E}_{Y|X}[Y \mid X = x] \,,$$

that is, the conditional expectation of the label Y given that $X = x$. Notice how the optimal model $\widehat{f}(x)$ depends only on the conditional expectation, and thus the conditional probability distribution $p_{Y|X}(y \mid x)$, giving more concrete justification to our earlier statement in Section 1.4 that in supervised learning we often ignore $p_X(x)$.

Note, in particular, if the label is a deterministic function of x, then seeing x once (i.e., drawing x as a training point x_i) is enough to determine $f(x)$. If there is noise in the labeling process, though, then one sample is not enough. However, in the training set there is typically only one observation $x_i = x$, and furthermore, in the test set we do not know the label and must estimate it. Therefore in k-nearest neighbors we replace $f(x) = \mathbb{E}[Y \mid X = x]$ with

$$f(x) = \text{Avg}[y_i \mid x_i \in N_k(x)] = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \,,$$

where we recall $N_k(x)$ consists of the k points from the training set T closest to x. The k-nearest neighbors algorithm is incorporating two approximations:

1. The expectation $\mathbb{E}_{Y|X}[Y \mid X = x]$ is approximated by a k-sample average over the training data.

2. The conditioning at the point $X = x$ is relaxed to conditioning on some region, the k-nearest neighbors in the training set, closest to the point x.

For large sample size N, the points in the k-nearest neighborhood are likely to be close to x, and as k gets larger the average will get more stable. In fact, under certain conditions on $p_{X,Y}(x, y)$, one can show that

$$\text{Avg}[y_i \mid x \in N_k(x_i)] \to \mathbb{E}_{Y|X}[Y \mid X = x] \text{ as } k, N \to \infty \text{ with } k/N \to 0\,.$$

However, we do not always have enough data points N to well approximate the above limits. In particular, as the dimension d increases, the number of sample points N needed to have k points close to each possible x increases rapidly, thus potentially leading to very bad approximations of $\mathbb{E}_{Y|X}[Y \mid X = x]$ (see Section 4.3 for more details).

The nearest neighbor model is good because it places very few assumptions on the underlying data generation process, encoded here by $p_{X,Y}(x, y)$. On the other hand, if we have prior knowledge about the data generation process, leveraging that knowledge and using a more structured class of models is preferred. The linear model described earlier is such a model. Recall the linear model is:

$$f(X; \theta) = \langle X, \theta \rangle = X^T \theta\,, \quad X, \theta : d \times 1\,,$$

where we have omitted the bias term but which can easily be incorporated or assumed to already be contained in X (as a constant dimension). Plugging this model into the true risk $R_{\text{true}}(f(\cdot; \theta))$ and minimizing the true risk with respect to θ (solved by differentiating with respect to θ and setting equal to zero) one obtains:

$$\widehat{\theta} = (\mathbb{E}_X[XX^T])^{-1}\mathbb{E}_{X,Y}[XY]\,.$$

Note the solution we obtained earlier for the empirical risk, given in (5), is the empirical version of the $\widehat{\theta}$ obtained here (in the empirical version we made the data points row vectors, here they are column vectors, hence the switching of the transposes). Either way, we do not condition on $X = x$. Rather we use the fact that we solving for an optimal linear model to average over the values of X and Y.

But what about classification? In the previous discussions, we assumed $\mathcal{Y} = \mathbb{R}$, but often we want to do classification and $\#(\mathcal{Y}) = M$, where M is the number of distinct classes. In this case the squared loss does not make sense. A standard choice is the 0-1 loss, which means that

$$\ell(y, f(x)) = \begin{cases} 0 & y = f(x) \\ 1 & y \neq f(x) \end{cases} \tag{10}$$

For now let us consider a general loss function, but we will come back to the 0-1 loss function shortly. Regardless of the choice of loss function, the true risk is:

$$R_{\text{true}}(f) = \mathbb{E}_{X,Y}[\ell(Y, f(X))]\,,$$

where we remind the reader that the expectation $\mathbb{E}_{X,Y}$ is taken over the joint probability density function of the data points X and the classes Y, $p_{X,Y}(x,y)$. Similar to the calculation concluded in (9), we can condition on the data points X:

$$\begin{aligned} R_{\text{true}}(f) = \mathbb{E}_{X,Y}[\ell(Y, f(X))] &= \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} \ell(y, f(x)) p_{X,Y}(x,y)\, dx \\ &= \int_{\mathcal{X}} \left(\sum_{y \in \mathcal{Y}} \ell(y, f(x)) p_{Y|X}(y \mid x) \right) p_X(x)\, dx \\ &= \mathbb{E}_X \left[\sum_{y \in \mathcal{Y}} \ell(y, f(X)) p_{Y|X}(y \mid X) \right] . \end{aligned}$$

Like before, from this calculation we again see it is sufficient to compute the optimal model, $\widehat{f}(x)$, point by point, meaning that:

$$\widehat{f}(x) = \arg\min_{y' \in \mathcal{Y}} \sum_{y \in \mathcal{Y}} \ell(y, y') p_{Y|X}(y \mid x) \,. \tag{11}$$

Now, if $\ell(y, f(x))$ is the 0-1 loss function (10), we can simplify (11) to

$$\widehat{f}(x) = \arg\min_{y' \in \mathcal{Y}} \left(1 - p_{Y|X}(y' \mid x)\right) ,$$

which itself is equivalent to

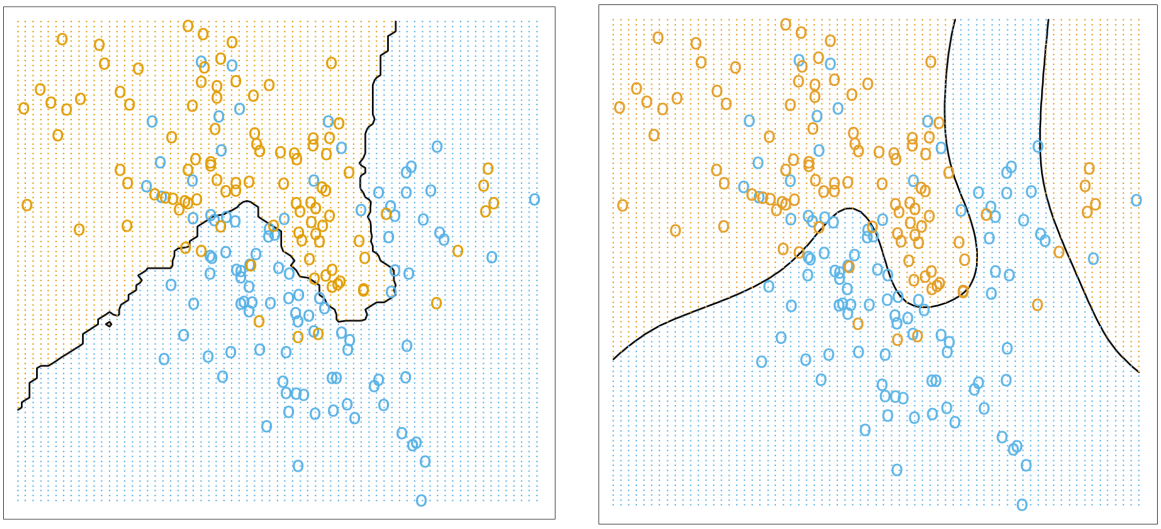
$$\widehat{f}(x) = \arg\max_{y' \in \mathcal{Y}} p_{Y|X}(y' \mid x) \,. \tag{12}$$

This model is called the *naive Bayes classifier*. It says, given a data point x, assign the most probable class using the conditional probability $p_{Y|X}(y \mid x)$; in retrospect, this is sort of obvious. The catch is that we don't know $p_{Y|X}(y \mid x)$, and so we must estimate it. The k-nearest neighbors algorithm is one way of doing so, and it can be pretty good; see Figure 12.

Remark 4.1. The Bayes classifier is not perfect because of uncertainty in the data generation process. The source of that uncertainty is similar to the noisy label model (4), but it is generated differently since here the labels are binary. Rather, the uncertainty comes from the way in which the two classes are distributed over $\mathbb{R}^2$. Here is a simpler example to give you an idea of what is going on. Consider data in $\mathbb{R}^2$ generated from a Gaussian mixture model consisting of two Gaussians, i.e.,

$$p_X(x) = \frac{1}{2} \left[\frac{1}{2\pi\sigma_1^2} e^{-\|x-\mu_1\|_2^2/2\sigma_1^2} + \frac{1}{2\pi\sigma_2^2} e^{-\|x-\mu_2\|_2^2/2\sigma_2^2} \right] .$$

The way that data is sampled from $p_{X,Y}(x) = p_X(x) p_{Y|X}(y \mid x)$ is the following. First we flip a coin. If it lands heads we sample $x \sim \mathcal{N}(\mu_1, \sigma_1)$ and we assign x the label $y =$ `blue` (-1).



(a) *The k-nearest neighbors classifier for* $k = 15$.

(b) *The naive Bayes classifier.*

Figure 12: *Comparison of the k-nearest neighbor classifier and the theoretical naive Bayes classifier. The k-nearest neighbor classifier does a good job of approximating the optimal Bayes decision boundary. Figures taken from [6].*

If it lands tails, we sample $x \sim \mathcal{N}(\mu_2, \sigma_2)$ and we assign x the label $y = \texttt{orange}$ $(+1)$. If the means μ_1, μ_2 are close enough and the standard deviations σ_1, σ_2 large enough, there will be significant overlap between the two distributions which creates uncertainty in the label of the point; see Figure 13. In our probabilistic framework, this means $p_{Y|X}(y \mid x)$ varies depending on the location of x, and in particular, $p_{Y|X}(\texttt{blue} \mid x) = p_{Y|X}(\texttt{orange} \mid x) = 1/2$ for x that are equidistant from μ_1 and μ_2 if $\sigma_1 = \sigma_2$.

Figure 13: *Two Gaussian mixture model in which there is significant overlap between the two Gaussians. Determining the label of new points sampled in the overlap region is difficult and creates uncertainty, even for the naive Bayes classifier with complete knowledge of the data generation process.*

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.

[6] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer-Verlag New York, 2nd edition, 2009.