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

# Lecture 05: MAP & k-Nearest Neighbors

January 22, 2019

*Lecturer: Matthew Hirn*

## 3.2 Bayes and maximum a posteriori

Recall from Section 1.3 we defined maximum likelihood estimation (MLE) as:

$$\widehat{\theta}_{\mathrm{MLE}} = \arg\max_{\theta \in \mathbb{R}^n} p_{X,Y}(T \mid \theta) = \arg\max_{\theta \in \mathbb{R}^n} p(T \mid \theta)\,,$$

where we recall $T = \{(x_i, y_i)\}_{i=1}^N$ is our training set. In other words, we maximized the probability of observing the particular training set $T$ that we have, given the model parameters $\theta$. On the other hand, a more intuitive and perhaps useful object might be $p(\theta \mid T)$, the probability of the model $\theta$ being correct given the training set $T$. Indeed, this is how we generally think of machine learning. We are given a training set $T$ and we pick a model $\theta$ that has the highest chance of describing the data. This is called the *maximum a posteriori* estimator:

$$\widehat{\theta}_{\mathrm{MAP}} = \arg\max_{\theta \in \mathbb{R}^n} p(\theta \mid T)\,. \tag{8}$$

The question then becomes, how do we compute $\widehat{\theta}_{\mathrm{MAP}}$? For this, we can use Bayes' Theorem:

$$p(\theta \mid T) \propto p(T \mid \theta) p(\theta)\,.$$

Notice that right hand side contains the term $p(T \mid \theta)$, which is what we used to compute the MLE model. But it also contains another, new term, $p(\theta)$. What is this term? It is referred to as the *prior distribution* on the parameters $\theta$. It encodes any prior knowledge we have about our model, or behavior that we want to build into our model. For example, we can place a Gaussian/normal prior on $\theta$, meaning that we assume each parameter $\theta(k)$ is distributed according to the normal distribution with mean zero and variance proportional to $\lambda^{-1}$:

$$p(\theta) = p(\theta \mid \lambda) = \prod_{k=1}^n \sqrt{\frac{\lambda}{2\pi}} e^{-\lambda|\theta(k)|^2} \quad \text{(normal prior)}\,.$$

We can also use a Laplace prior:

$$p(\theta) = p(\theta \mid \lambda) = \prod_{k=1}^n \frac{\lambda}{2} e^{-\lambda|\theta(k)|} \quad \text{(Laplace prior)}\,.$$

Let us now see how the $\widehat{\theta}_{\text{MAP}}$ model is related to regularized functional models, and in particular ridge regression for the normal prior and LASSO for the Laplace prior. Using (8) we have:

$$\begin{aligned}\widehat{\theta}_{\text{MAP}} &= \arg\max_{\theta\in\mathbb{R}^n} p(\theta \mid T) \\ &= \arg\max_{\theta\in\mathbb{R}^n} p(T \mid \theta)p(\theta) \\ &= \arg\max_{\theta\in\mathbb{R}^n} (\log p(T \mid \theta) + \log p(\theta))\end{aligned}$$

Recall from (3) that if the $\{x_i\}_{i=1}^N$ are sampled iid (independently and identically distributed) from $\mathcal{X}$, then we can decompose $\log p(T \mid \theta)$ as:

$$\widehat{\theta}_{\text{MAP}} = \arg\max_{\theta\in\mathbb{R}^n} \left( \sum_{i=1}^N \log p_X(x_i \mid \theta) + \sum_{i=1}^N \log p_{Y|X}(y_i \mid x_i, \theta) + \log p(\theta) \right) .$$

Furthermore, in the case of supervised learning we will often discard the first term involving $p_X$, and, as we saw in Section 2.2, taking

$$p_{Y|X}(y \mid x, \theta) = e^{-|y-f(x;\theta)|^2/2\sigma^2} ,$$

is a reasonable choice. We then have:

$$\widehat{\theta}_{\text{MAP}} = \arg\min_{\theta\in\mathbb{R}^n} \left( \frac{1}{2\sigma^2} \sum_{i=1}^N |y_i - f(x_i;\theta)|^2 - \log p(\theta) \right) .$$

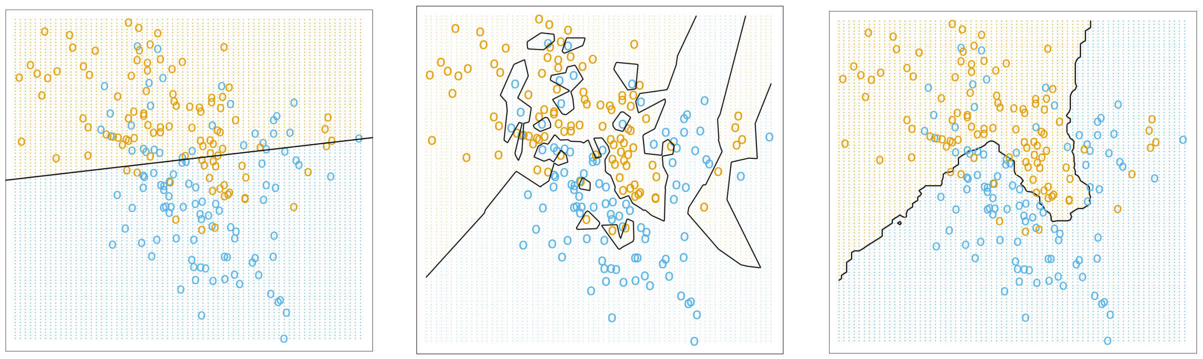
We thus see that $\log p(\theta)$ serves as the regularization term on the parameters $\theta$. In particular, if we use the Gaussian/normal prior, we obtain:

$$\widehat{\theta}_{\text{MAP}} = \arg\min_{\theta\in\mathbb{R}^n} \left( \frac{1}{2\sigma^2} \sum_{i=1}^N |y_i - f(x_i;\theta)|^2 + \lambda \sum_{k=1}^n |\theta(k)|^2 \right) ,$$

which is precisely ridge regression if $f(x;\theta) = \langle (1,x), \theta \rangle$. Similarly, if we use the Laplace prior, we obtain LASSO.

## 3.3 $k$-Nearest neighbors and local methods

Figure 7c illustrates the need for a nonlinear classifier, capable of learning a nonlinear decision boundary as opposed to a straight line. A polynomial method could do better, but this still imposes a modeling assumption on the underlying data generation process. In order to avoid such assumptions, we instead turn to another type of method, *$k$-nearest neighbors*, which is a local method.



**(a)** *Linear regression* **(b)** *k-nearest neighbors with* $k = 1$ **(c)** *k-nearest neighbors with* $k = 15$

**Figure 10:** *Two dimensional data with two classes, orange (+1) and blue (-1). Three different classifiers and their respective decision boundaries. Figure taken from [6].*

Let $N_k(x)$ denote $k$ nearest of neighbors of $x$, in the Euclidean distance, from the set of training points $\{x_1, \ldots, x_N\} \subset \mathbb{R}^d$. The $k$-nearest neighbor model is:

$$f(x; k) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \, .$$

Figure 10 illustrates the model for $k = 1$ and $k = 15$, and compares to the linear classifier from Figure 7c. In the figure we see that the $k$-nearest neighbors classifiers make significantly fewer mis-classifications on the training set than the linear regression classifier. However, classification rate on the training set is not the sole criterion for the quality of a model, as we saw in our discussion on regularization. Indeed models must generalize well to unseen points. The $k = 1$ nearest neighbor model in fact makes no mis-classifications on the training set, since it simply assigns to each training point its own label. But the decision boundary of the 1-nearest neighbor classifier is extremely irregular and likely to make mistakes on test data. The $k = 15$ nearest neighbor classifier, on the other hand, has a semi-regular boundary that is likely to generalize better than either the 1-nearest neighbor classifier (because it is too rough) or the linear regression classifier (because it is too smooth). This discussion is hinting at the bias-variance tradeoff in machine learning, which we will make more precise in Section 4. It also illustrates that the $k = 1$ model is, in some sense, more complex than the $k = 15$ model. In the extreme case, $k = N$, all points are classified the same, and so this is the simplest model. In fact, even though there is only one parameter, $k$, that we set, the effective number of parameters or complexity of the $k$-nearest neighbor model is $N/k$.

Note that because the 1-nearest neighbor classifier will always make zero errors on the training set, we require a different method by which to select $k$. We instead use cross-validation, which requires a validation set separate from the training set, but for which we still know the correct labels. In fact, many hyper-parameters, including $k$ in $k$-nearest neighbors and $\lambda$ in regularized linear models, are selected through cross-validation. The way

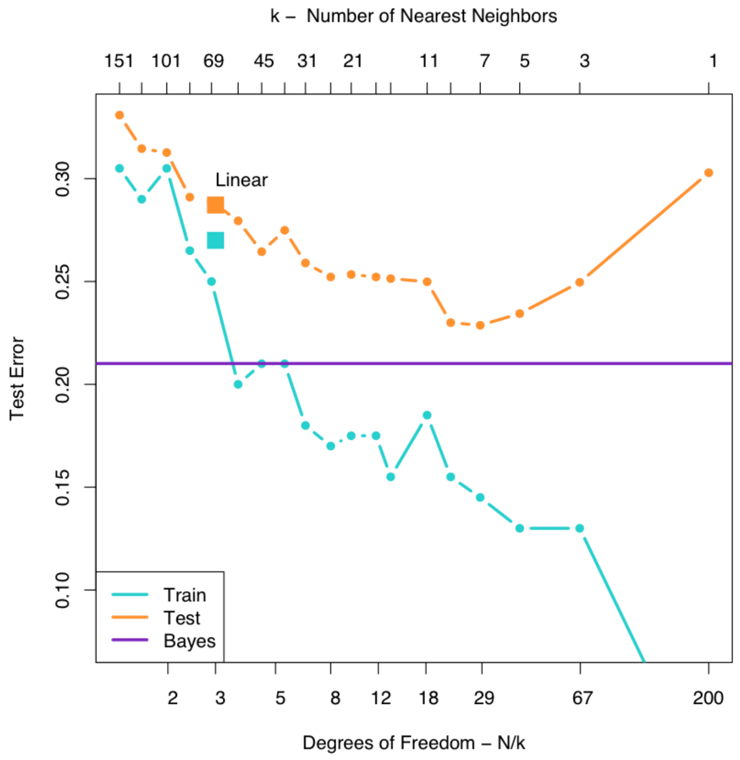


**Figure 11:** *$k$-nearest neighbors classifier for $k$ decreasing from left to right, evaluated on the training set and the test (validation) set. The models with $k \approx 10$ perform the best on the test set, even though the training error is essentially decreasing as $k \to 1$.*

it works for $k$-nearest neighbors is that we use the original training set $T$ to define the neighborhoods of each point and the model $f(x; k)$, but we then evaluate this model on the validation set and compute the average error on the validation set. The $k$ with the lowest average error on the validation set is the model we use on the test set. See Figure 11 for an illustration using $k$-nearest neighbors. The more general principle, which includes $k$-nearest neighbors and the regularized linear regularization, is based on the bias variance tradeoff, which will be discussed in the next section.

## 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.