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

Lecture 10: Dictionaries and Kernels

February 3, 2020

Lecturer: Matthew Hirn

5 Towards deep learning

5.1 Dictionaries and kernels

One way to incorporate nonlinearity is through dictionaries and kernels. We give a brief overview here. A *dictionary* takes in a data point $x \in \mathcal{X}$ and maps it into a Hilbert space $\mathcal{H}$. Often the Hilbert space is usually $\mathcal{H} = \mathbb{R}^m$, so let us assume that is the case here. The resulting representation of x is denoted:

$$\Phi : \mathcal{X} \to \mathbb{R}^m \, , \quad x \mapsto \Phi(x) = (\phi_k(x))_{k=1}^m \, , \quad \phi_k : \mathcal{X} \to \mathbb{R} \, .$$

Now with the representation $\Phi(x)$, we can apply linear regression but to the representation:

$$f(x; \theta) = \langle \Phi(x), \theta \rangle = \sum_{k=1}^m \theta(k) \phi_k(x) \, . \tag{18}$$

More generally, any algorithm that only depends upon $\langle x, \theta \rangle$ can be recast be replacing x with $\Phi(x)$. The resulting model class $\mathcal{F}$ or hypothesis class $\mathcal{P}$ thus depends strongly on the choice of dictionary Φ. The constituent elements of $\Phi(x)$, the functions $\phi_k(x)$, are often referred to as "features." Because of the importance of these features, feature engineering has been and continues to be an important topic in machine learning, although "hand crafted" features are becoming less popular. Nevertheless, if one can come up with a dictionary $\Phi(x)$ for which the label $y(x)$ is a linear function of $\Phi(x)$, then the linear model over the dictionary given by (18) will work. Note that the dimension of the problem is now m instead of d, but from Section 4.3 we know that linear models circumvent the curse of dimensionality in that the number of training points need only grow linearly with the dimension. In this case that means we need $N = O(m)$ training points. If m is not too much larger than d, then we will be in an favorable situation.

Kernels and deep learning do not specify the features directly, but they do so in different ways. To understand kernel methods, let us define a kernel $K(x, x')$ in terms of a dictionary $\Phi(x)$ as:

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle \, .$$

The *kernel* $K(x, x')$ measures the similarity between x and x' through the lens of Φ; the larger $K(x,x')$, the more similar x and x'. A kernel regression computes:

$$f(x;\alpha) = \sum_{i=1}^{N} \alpha(i) K(x, x_i)\,, \quad \alpha \in \mathbb{R}^N\,, \tag{19}$$

where we have assumed the kernel is symmetric, meaning $K(x,x') = K(x',x)$. In other words, $f(x;\alpha)$ predicts the label of x by comparing x to each training point x_i through $K(x,x_i)$ and taking a weighted sum of the resulting similarity measures. Notice that we can rewrite (19) as:

$$\begin{aligned} f(x;\alpha) = \sum_{i=1}^{N} \alpha(i) K(x,x_i) &= \sum_{i=1}^{N} \alpha(i) \langle \Phi(x), \Phi(x_i)\rangle \\ &= \sum_{i=1}^{N} \alpha(i) \sum_{k=1}^{n} \phi_k(x)\phi_k(x_i) \\ &= \sum_{k=1}^{m} \left[\sum_{i=1}^{N} \alpha(i)\phi_k(x_i) \right] \phi_k(x) \\ &= \sum_{k=1}^{m} \theta(k)\phi_k(x)\,, \end{aligned} \tag{20}$$

where we have set

$$\theta(k) = \sum_{i=1}^{N} \alpha(i)\phi_k(x_i)\,.$$

Since $\alpha \in \mathbb{R}^N$ and $\theta \in \mathbb{R}^m$ are learned from the training data, we see that (18) and (19) are equivalent.

Notice, though, that (19) can be defined for any kernel K, not just those for which $K(x,x') = \langle \Phi(x), \Phi(x')\rangle$. On the other hand, many algorithms depend on just the inner product $\langle x, x'\rangle$ to measure the similarity between two data points. If we want to replace these inner products with $K(x,x')$ and still have the algorithm behave well, we need to at least know that $K(x,x') = \langle \Phi(x), \Phi(x')\rangle$ for some Φ, but we do not need to know Φ. This is the key observation of *kernel learning* [8]. Kernel learning allows one to implicitly use very complicated representations $\Phi(x)$ via what are often simple kernels $K(x,x')$. For example, the polynomial kernel

$$K(x,x') = (\langle x, x'\rangle + c)^m\,,$$

implicitly defines the polynomial model class of degree m. Another example is the Gaussian kernel

$$K(x,x') = e^{-\|x-x'\|_2^2/2\sigma^2}\,,$$

which implicitly defines a nonlinear infinite dimensional dictionary $\Phi(x)$!

Because of this observation and the usefulness of kernels and the simplicity of incorporating them into machine learning, there is a well developed mathematical theory for determining when a kernel $K(x, x')$ implicitly defines a dictionary $\Phi(x)$ and thus can be written as $K(x, x') = \langle \Phi(x), \Phi(x') \rangle$. The summary of this theory is that K must be a *reproducing kernel* and thus generate a *reproducing kernel Hilbert space (RKHS)*. This is also not easy to verify directly, but it turns out that K is a reproducing kernel if and only if it is *postitive semi-definite*, which means that for any N and any $\{x_i\}_{i=1}^N \subset \mathcal{X}$ and any $c \in \mathbb{R}^N$,

$$\sum_{i,j=1}^{N} c(i)c(j)K(x_i, x_j) \geq 0\,.$$

For more details we refer the reader to [8]; see also the course CMSE 820 (my old version here or newer versions by Prof. Yuying Xie), which has thus far covered this topic every year.

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.

[7] J. Hartlap, P. Simon, and P. Schneider. Why your model parameter confidences might be too optimistic - unbiased estimation of the inverse covariance matrix. *Astronomy and Astrophysics*, 464(1):399–404, 2007.

[8] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. The MIT Press, 2002.