Kernel Regression and Neural Tangent Kernels

Bhaskar Mukhoty

October 28, 2026

1 Kernel Regression via Least Squares

Let's start with the least-squares regression with n training data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ given in matrix notation as $(X, \mathbf{y})$, where $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]^\top \in \mathbb{R}^{n\times d}$ holds data-points $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{y} = [y_1, \cdots, y_n]^\top \in \mathbb{R}^n$ be the corresponding vectors responses. The least-square loss incurred by the model $\mathbf{w} \in \mathbb{R}^d$ is given as:

$$l(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \langle \mathbf{x}_i, \mathbf{w}\rangle)^2 = \|\mathbf{y} - X\mathbf{w}\|_2^2 \tag{1}$$

Setting gradient of the loss to zero,

$$\nabla l(\mathbf{w})|_{\mathbf{w}=\mathbf{w}^\star} = -2X^\top(\mathbf{y} - X\mathbf{w}^\star) = 0 \quad \iff \quad X^\top X\mathbf{w}^\star = X^\top \mathbf{y} \tag{2}$$

Assuming $X^\top X$ is invertible (i.e., X has d independent columns):

$$\mathbf{w}^\star = (X^\top X)^{-1} X^\top \mathbf{y} \tag{3}$$

For a new point $\hat{\mathbf{x}}$, the prediction is made as:

$$\hat{y} = \langle \hat{\mathbf{x}}, \mathbf{w}^\star\rangle = \hat{\mathbf{x}}^\top (X^\top X)^{-1} X^\top \mathbf{y} \tag{4}$$

This is the standard treatment of the problem where given training data $(X, \mathbf{y})$ we learn $\mathbf{w}^\star$ and use it for prediction. However, there is an **alternate way** of doing it:

Recall that null space and row space of X are orthogonal, $\text{Null}(X) \perp \text{Col}(X^\top)$. Let $\mathbf{w} = \mathbf{w}_n + \mathbf{w}_r$, so that, $\mathbf{w}_n \in \text{Null}(X), \mathbf{w}_r \in \text{Col}(X^\top)$, then we will have, $X\mathbf{w} = X\mathbf{w}_r$, concluding that one can work with $\mathbf{w} \in \text{Col}(X^T)$,

$$\mathbf{w} = X^T \boldsymbol{\alpha} \tag{5}$$

So that, we write,

$$l(\boldsymbol{\alpha}) = \sum_{i=1}^{n} (y_i - \langle \mathbf{x}_i, X^\top \boldsymbol{\alpha}\rangle)^2 = \|\mathbf{y} - XX^\top \boldsymbol{\alpha}\|_2^2 = \|\mathbf{y} - K\boldsymbol{\alpha}\|_2^2 \tag{6}$$

Observe, **linear kernel** $K := XX^T \in \mathbb{R}^{n\times n}$ holds all pair of inner products, i.e., $K_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j\rangle$. Setting gradient to zero we have, and assuming K to be invertible (X is full row-rank):

$$\begin{aligned} \nabla l(\boldsymbol{\alpha}) &= -2K(\mathbf{y} - K\boldsymbol{\alpha}) = 0 \\ \boldsymbol{\alpha}^\star &= K^{-1}\mathbf{y} \end{aligned} \tag{7}$$

For prediction on a new input $\hat{\mathbf{x}}$, we have,

$$\hat{y} = \langle \hat{\mathbf{x}}, X^\top \boldsymbol{\alpha}^\star\rangle = \hat{\mathbf{x}}^\top X^\top \boldsymbol{\alpha}^\star = \hat{\mathbf{x}}^\top X^\top K^{-1}\mathbf{y} = \mathbf{k}_X(\hat{\mathbf{x}})^\top K^{-1}\mathbf{y} \tag{8}$$

Here use the notation $\mathbf{k}_X(\hat{\mathbf{x}}) = X\hat{\mathbf{x}} \in \mathbb{R}^n$, which gives inner product of $\hat{\mathbf{x}}$ with all training points. Note that, to make a prediction datapoints interacts only through inner-products. This is crucial in **Kernel regression**, in which, we want to solve:

$$l(\mathbf{w}) = \sum_{i=1}^{n}(y_i - \langle\phi(\mathbf{x}_i), \mathbf{w}\rangle)^2 = \|\mathbf{y} - \Phi\mathbf{w}\|_2^2 \quad \text{where, } \Phi = \begin{bmatrix}\phi(\mathbf{x}_1)^\top \\ \phi(\mathbf{x}_2)^\top \\ \vdots \\ \phi(\mathbf{x}_n)^\top\end{bmatrix} \tag{9}$$

$$\text{equivalently,} \quad l(\boldsymbol{\alpha}) = \|\mathbf{y} - \Phi\Phi^T\boldsymbol{\alpha}\|_2^2 \quad \text{with,} \mathbf{w} = \Phi^\top\boldsymbol{\alpha} \tag{10}$$

For $\phi : \mathbb{R}^d \to \mathcal{H}$ that maps to an infinite dimensional space, so that, the feature matrix Φ is difficult to store. However, the Kernel matrix $K = \Phi\Phi^\top \in \mathbb{R}^{n\times n}$, with $K_{i,j} = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_i)\rangle_{\mathcal{H}}$ can be computed and stored. So that, we have,

$$\boldsymbol{\alpha}^\star = K^{-1}\mathbf{y}, \quad \text{and,} \quad \hat{y} = \mathbf{k}_X(\hat{\mathbf{x}})^\top K^{-1}\mathbf{y} \tag{11}$$

1.1 Neural Tangent Kernel

Here we consider a neural network $f_\theta(\mathbf{x})$ parameterized by θ that works on the input $\mathbf{x}$. Let the network is initialized at θ^0, using first-order approximation, we may write:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) \approx f_{\boldsymbol{\theta}^0}(\mathbf{x}) + \langle\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^0}(\mathbf{x}), \boldsymbol{\theta} - \boldsymbol{\theta}^0\rangle \tag{12}$$

which is a affine function in $\boldsymbol{\theta}$, let, $\phi(\mathbf{x}) = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^0}(\mathbf{x})$ minimizing the least-squares loss we may write:

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{n}(y_i - f_{\boldsymbol{\theta}^0}(\mathbf{x}_i) - \langle\phi(\mathbf{x}_i), \boldsymbol{\theta} - \boldsymbol{\theta}^0\rangle)^2$$

where, $\phi(\mathbf{x}) = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^0}(\mathbf{x})$. To express this problem in the kernel formulation, let

$$\mathbf{u} = \boldsymbol{\theta} - \boldsymbol{\theta}^0, \qquad r_i = y_i - f_{\boldsymbol{\theta}^0}(\mathbf{x}_i), \quad \mathbf{r} = [r_1, \cdots, r_n]^T, \qquad \Phi = \begin{bmatrix}\phi(\mathbf{x}_1)^\top \\ \phi(\mathbf{x}_2)^\top \\ \vdots \\ \phi(\mathbf{x}_n)^\top\end{bmatrix} \tag{13}$$

The loss can be written as,

$$l(\mathbf{u}) = \|\mathbf{r} - \Phi\mathbf{u}\|_2^2 \tag{14}$$

Following the same argument as in kernel least-squares regression, we may choose a minimizer in the row space of Φ, so that, $\mathbf{u} = \Phi^\top\boldsymbol{\alpha}$, and $K = \Phi\Phi^\top \in \mathbb{R}^{n\times n}$, consequently,

$$l(\boldsymbol{\alpha}) = \|\mathbf{r} - \Phi\Phi^\top\boldsymbol{\alpha}\|_2^2 = \|\mathbf{r} - K\boldsymbol{\alpha}\|_2^2 \tag{15}$$

Thus, the NTK measures the inner product between the parameter gradients of the network outputs at two inputs. Since, $\boldsymbol{\alpha}^\star = K^{-1}\mathbf{r}$, for a new point $\hat{\mathbf{x}}$ we have,

$$\begin{aligned} \hat{r} &= \mathbf{k}_X(\hat{\mathbf{x}})^\top K^{-1}\mathbf{r} \\ \text{or,} \quad \hat{y} &= f_{\theta^0}(\hat{\mathbf{x}}) + \mathbf{k}_X(\hat{\mathbf{x}})^\top K^{-1}(\mathbf{y} - \mathbf{f}_0) \end{aligned} \tag{16}$$

where $\mathbf{f}_0 = [f_{\theta^0}(\mathbf{x}_1), \cdots, f_{\theta^0}(\mathbf{x}_n)]^T$ makes slight abuse the notation to denote the vector of network predictions on training data points. The expression shows that the predictions of the fitted linearized network can be approximated using only the initial network outputs and pairwise NTK evaluations, without explicitly performing the parameter update.

Connection to gradient descent. The kernel formulation also describes the evolution of predictions during gradient descent on the linearized model. Under gradient flow, we can write using Eqn. 14:

$$\frac{d\mathbf{u}(t)}{dt} = -\nabla_{\mathbf{u}} l(\mathbf{u}(t)) = \Phi^\top (\mathbf{r} - \Phi\mathbf{u}(t)), \tag{17}$$

Using linear approximation of Eqn. 12, and applying it on the training points, we have,

$$\hat{\mathbf{f}}(t) = \mathbf{f}_0 + \Phi\mathbf{u}(t) \tag{18}$$

Differentiating gives:

$$\frac{d\hat{\mathbf{f}}(t)}{dt} = \Phi\frac{d\mathbf{u}(t)}{dt} = \Phi\Phi^\top (\mathbf{r} - \Phi\mathbf{u}(t)) = K(\mathbf{y} - \hat{\mathbf{f}}(t)) \tag{19}$$

Thus, the evolution of the training predictions is governed entirely by the NTK K. In the linearized model, K is fixed because it is evaluated at initialization. For a sufficiently wide neural network, under suitable architectural and scaling assumptions, the NTK can converge to a deterministic limiting kernel as the width tends to infinity. In this regime, the network's training dynamics can often be understood through kernel methods [Jacot et al., 2018]

References

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.