Lecture 3: Kernel Regression

Adityanarayanan Radhakrishnan

Edited by: Max Ruiz Luyten, George Stefanakis, Cathy Cai

January 14, 2023

1 Introduction

Thus far, we have developed the linear regression framework, which identifies a line of best fit to map from samples to labels. However, in modern datasets, e.g. image classification, it is rarely the case that there exists an effective linear mapping from samples to labels. Hence, we now extend from linear regression to kernel regression to learn a nonlinear mapping from samples to labels. We conclude this lecture by presenting a connection between solving kernel regression and training the last layer of an infinitely wide neural network. Such a connection will lead naturally into the development of Neural Network Gaussian Processes (NNGP) and the Neural Tangent Kernel.

2 From Linear to Kernel Regression

Consider the empirical risk minimization framework from the previous lecture, which was used to learn a mapping from training samples $\{x^{(i)}\}_{i=1}^n \subset \mathbb{R}^d$ to labels $\{y^{(i)}\}_{i=1}^n \subset \mathbb{R}$. Recall that given a parameterized function class (e.g. the set of linear functions from $\mathbb{R}^d$ to $\mathbb{R}$), we selected $\hat{f}(x) = \hat{w}x$ where $\hat{w}$ by minimizing:

$$\mathcal{L}(w) = \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - wx^{(i)})^2$$

In order to extend linear regression into nonlinear regression, we will simply apply a fixed nonlinear transform to samples x before performing linear regression. Formally, we will consider the set of class of functions

$$\mathcal{F} = \{f : \mathbb{R}^d \to \mathbb{R} \ ; \ f(x) = \langle w, \psi(x)\rangle_{\mathcal{H}} \ , \ \psi : \mathbb{R}^d \to \mathcal{H} \ , \ w \in \mathcal{H}\} \ ;$$

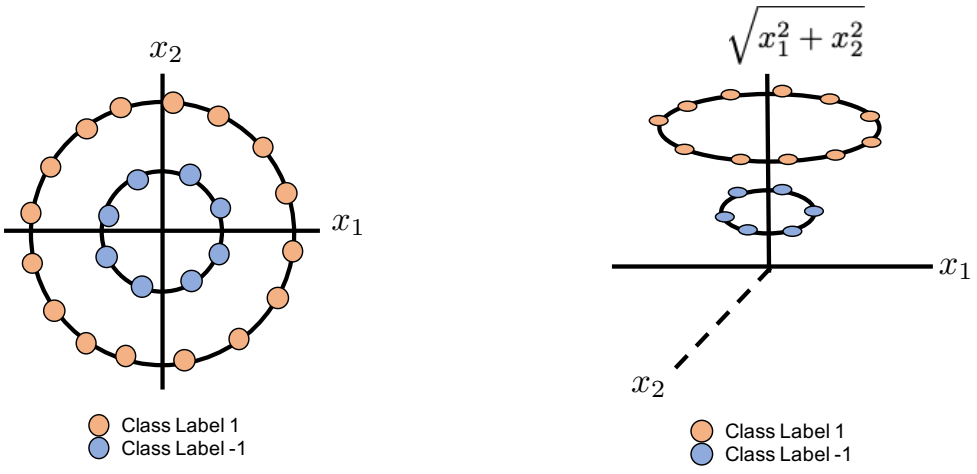
where $\mathcal{H}$ is a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and ψ is a nonlinear feature map.[1] The following example in which $\mathcal{H} = \mathbb{R}^3$ provides intuition as to why such an approach can be effective.

Example 1. *Consider data drawn from the 2D distribution shown in Fig. 1a, where each sample lies on one of two concentric circles corresponding to different classes. Note that in 2 dimensions, the data is not linearly separable, i.e. there is no line we can draw that separates the orange from the blue points. On the other hand, as the radius of each circle is a key feature for classifying the points, consider the following feature map $\psi : \mathbb{R}^2 \to \mathbb{R}^3$ given by*

$$\psi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \\ \sqrt{x_1^2 + x_2^2} \end{bmatrix}.$$

Note that the added feature corresponds to the radius of the circle on which a point lies. In Fig. 1b, we visualize the data after applying the map ψ to each of our data points. While the data were not linearly separable in 2 dimensions, they are now linearly separable in three, i.e. we can draw a plane that separates the orange and blue points.

[1] Recall that a Hilbert space is a complete inner product space.



(a) (b)

Figure 1: A simple demonstration of the effectiveness of using a linear predictor on a nonlinear transformation of features. (a) A dataset in where the two classes (corresponding to orange and blue points) are not linearly separable, i.e. there is no line separating the two classes. (b) Adding the feature $\sqrt{x_1^2 + x_2^2}$ to the data makes the transformed features linearly separable in 3 dimensions.

As demonstrated in the example, by selecting an appropriate feature transformation, we can still construct an effective predictor even while using a linear function class. Thus, a natural question is how to select an appropriate feature transformation for any given dataset. As we will demonstrate in the following lecture, when there is no prior knowledge about useful features, it is often beneficial to consider random feature maps into a Hilbert space. Nevertheless, we will return to answering this fundamental question after first developing the fundamentals for finding a solution to linear regression after applying a general feature map, $\psi : \mathbb{R}^d \to \mathcal{H}$. Note that even though $\mathcal{H}$ can be infinite dimensional, we can remarkably still solve linear regression in such a space by utilizing the following Representer theorem [3].

2.1 Representer Theorem

Consider performing linear regression to map from samples transformed by a feature map ψ, e.g. $\{\psi(x^{(i)})\}_{i=1}^n \subset \mathcal{H}$ to labels $\{y^{(i)}\}_{i=1}^n \subset \mathbb{R}$. Suppose $\mathcal{H}$ is a finite dimensional Hilbert space with the usual ℓ_2 inner product, then we can follow the analysis from the previous lecture to solve for the minimum norm solution. In particular, recall from Theorem 1 of Lecture 2 that using gradient descent initialized at $w^{(0)} = \mathbf{0}$ to minimize the loss

$$\mathcal{L}(w) = \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - wx^{(i)})^2 \quad ;$$

led to $w^* = yX^\dagger$, which was the minimum norm solution. We now present a key property of this minimum norm solution.

Proposition 1. *Let $\{x^{(i)}\}_{i=1}^n \subset \mathbb{R}^d$ and $\{y^{(i)}\}_{i=1}^n \subset \mathbb{R}$. Then there exist $\{\alpha_i\}_{i=1}^n \subset \mathbb{R}$, such that the minimum ℓ_2 norm minimizer, w^*, for the loss:*

$$\mathcal{L}(w) = \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - wx^{(i)})^2 \quad ;$$

has the form

$$w^* = \sum_{i=1}^{n}\alpha_i {x^{(i)}}^T \quad ;$$

Proof. The fact that $w^* = yX^\dagger$ is the minimum ℓ_2 norm solution is shown in the homework. Naturally, we can conclude the result by noting that the span of $yX^\dagger$ is equal to the span of X. However, here we will also show that gradient descent actually preserves this property. We use induction to show that for any $t \in \mathbb{N}$, the solution given by gradient descent at timestep t has the form:

$$w^{(t)} = \sum_{i=1}^{n} \alpha_i^{(t)} {x^{(i)}}^T$$

The statement clearly holds for $w^{(0)}$ (e.g. $\alpha_i = 0$ for all i). Thus, assume it is true for time t. Then for $t+1$, we have:

$$\begin{aligned}
w^{(t+1)} &= w^{(t)} + \eta(y - w^{(t)}X)X^T \\
&= w^{(t)} + \eta \sum_{i=1}^{n} \beta_i^{(t)} {x^{(i)}}^T \quad (\beta_i^{(t)} \text{ is the } i^{th} \text{ coordinate of } y - w^{(t)}X \in \mathbb{R}^{1\times n}) \\
&= \sum_{i=1}^{n} (\alpha_i^{(t)} + \eta\beta_i^{(t)}) {x^{(i)}}^T \\
&= \sum_{i=1}^{n} \alpha_i^{(t+1)} {x^{(i)}}^T
\end{aligned}$$

Hence, $w^{(t+1)}$ has the desired form and induction is complete. □

Now if $\mathcal{H}$ is finite dimensional, we can repeat this previous analysis by substituting X with the transformed sample matrix $\psi(X) = [\psi(x^{(1)})| \dots |\psi(x^n)]$. A key aspect of the previous analysis is that it is extendable to the case where $\mathcal{H}$ is a general Hilbert space. In particular, Proposition 1 is generalized as follows.

Theorem 1 (Representer Theorem)**.** *Let $\mathcal{H}$ be a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Let $\{\psi(x^{(i)})\}_{i=1}^n \subset \mathcal{H}$ and $\{y^{(i)}\}_{i=1}^n \subset \mathbb{R}$. Then there exist $\{\alpha_i\}_{i=1}^n \subset \mathbb{R}$, such that the minimum $\mathcal{H}$-norm minimizer, w^*, for the loss:*

$$\mathcal{L}(w) = \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \langle w, \psi(x^{(i)}) \rangle_{\mathcal{H}})^2 \quad ; \tag{1}$$

lies in the span of the samples $\{\psi(x^{(i)})\}_{i=1}^n$, i.e.

$$w^* = \sum_{i=1}^{n} \alpha_i \psi(x^{(i)}) \quad ;$$

Proof. We prove the result by first showing that all minimizers of the loss differ only by a term that is orthogonal to the transformed training samples, and then, we will use the Pythagorean theorem to show that the solution in the span of the transformed training samples has minimum norm. In particular, consider the orthogonal decomposition of any $\tilde{w} \in \mathcal{H}$ onto the space spanned by $\psi(x^{(i)})$ and its complement. In particular, there exists some orthonormal basis $\{\phi_i\}_{i=1}^n \subset \mathcal{H}$ for $\{\psi(x^{(i)})\}_{i=1}^n$ and some $v \in \mathcal{H}$ orthogonal to all ϕ_i such that:

$$\tilde{w} = \sum_{i=1}^{n} \beta_i \phi_i + v$$

We thus have that:

$$\begin{aligned}\mathcal{L}(\tilde{w}) &= \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \langle \tilde{w}, \psi(x^{(i)})\rangle_{\mathcal{H}})^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \langle \sum_{i=1}^{n}\beta_i\phi_i + v, \psi(x^{(i)})\rangle_{\mathcal{H}})^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \langle \sum_{i=1}^{n}\beta_i\phi_i, \psi(x^{(i)})\rangle_{\mathcal{H}})^2 \\ &= \mathcal{L}(w^*)\end{aligned}$$

Hence, the loss is unaffected by adding a term orthogonal to the span of the $\psi(x^{(i)})$. Moreover, by the Pythagorean theorem:

$$\|\tilde{w}\|_{\mathcal{H}}^2 = \|\sum_{i=1}^{n}\beta_i\phi_i + v\|_{\mathcal{H}}^2 = \|\sum_{i=1}^{n}\beta_i\phi_i\|_{\mathcal{H}}^2 + \|v\|_{\mathcal{H}}^2 \geq \|w^*\|_{\mathcal{H}}^2$$

Hence, the minimum $\mathcal{H}$-norm minimizer of the loss admits the desired representation. □

For any feature map $\psi : \mathbb{R}^d \to \mathcal{H}$, we can thus solve linear regression in a Hilbert space by first solving for the coefficients α_i and utilizing the representation for the minimum norm solution given in Theorem 1. This procedure is referred to as kernel regression.

2.2 Kernel Regression and the Kernel Trick

We will now use the result of Theorem 1 to convert the seemingly intractable problem of minimizing the loss in Eq. (1) to solving a finite dimensional linear regression problem. In particular, we substitute $w = \sum_{i=1}^{n}\alpha_i\psi(x^{(i)})$ to simplify the loss in Eq. (1) as follows:

$$\begin{aligned}\mathcal{L}(w) &= \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \langle w, \psi(x^{(i)})\rangle_{\mathcal{H}})^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \langle \sum_{j=1}^{n}\alpha_j\psi(x^{(j)}), \psi(x^{(i)})\rangle_{\mathcal{H}})^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \begin{bmatrix}\alpha_1 & \alpha_2 & \dots & \alpha_n\end{bmatrix}\begin{bmatrix}\langle\psi(x^{(1)}), \psi(x^{(i)})\rangle_{\mathcal{H}} \\ \langle\psi(x^{(2)}), \psi(x^{(i)})\rangle_{\mathcal{H}} \\ \vdots \\ \langle\psi(x^{(n)}), \psi(x^{(i)})\rangle_{\mathcal{H}}\end{bmatrix}\right)^2 \end{aligned} \tag{2}$$

Therefore, instead of minimizing $\mathcal{L}(w)$ over all w, we minimize the loss $\mathcal{L}$ with respect to the parameters $\{\alpha_i\}_{i=1}^n$. Theorem 1 implies that identifying these α_i's will yield the minimum $\mathcal{H}$-norm solution that minimizes the loss in Eq. (1).

Importantly, Eq. (2) implies that we need only know the inner products $\langle\psi(x^{(i)}), \psi(x^{(j)})\rangle_{\mathcal{H}}$ for all $i, j \in [n]$ to perform linear regression in a Hilbert space. Moreover, we do not even need to know the map ψ, but rather the functional that yields the required inner products. Namely, we only need some function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that $K(x, \tilde{x}) = \langle\psi(x), \psi(\tilde{x})\rangle_{\mathcal{H}}$. This is formalized by the notion of kernels.

Definition 1 (Kernel)**.** *Given nonempty set $\mathcal{X}$, a **kernel** is a symmetric continuous function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.*

Note that requiring K to have this inner product form introduces constraints, e.g. $K(x, x) \geq 0$. Thus, we will consider kernels that satisfy the positive semi-definite constraint as defined below.

Definition 2 (Positive semi-definite kernel)**.** *Given nonempty set $\mathcal{X}$, a kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is* ***positive semi-definite*** *iff for any $\{x^{(i)}\}_{i=1}^n \subset \mathcal{X}$ and for any $\{c_i\}_{i=1}^n \subset \mathbb{R}$,*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K(x^{(i)}, x^{(j)}) \geq 0.$$

Although out of scope for this course, we note that there are easier conditions for verifying if a given function is a kernel, e.g. Bochner's theorem [9]. The following proposition (proved in the homework) implies that those kernels induced by a feature map are naturally positive semi-definite kernels.

Proposition 2. *Let $\mathcal{H}$ be a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that $K(x, \tilde{x}) = \langle \psi(x), \psi(\tilde{x}) \rangle_{\mathcal{H}}$ for $\psi : \mathbb{R}^d \to \mathcal{H}$. Then K is a positive semi-definite kernel.*

We here provide some classical examples of positive semi-definite kernels.

1. **Linear Kernel.** $K(x, \tilde{x}) = x^T \tilde{x}$.
2. **Gaussian (RBF) Kernel.** $K(x, \tilde{x}) = \exp\left(-L\|x - \tilde{x}\|_2^2\right)$ for $L \in \mathbb{R}_+$.
3. **Laplace Kernel.** $K(x, \tilde{x}) = \exp\left(-L\|x - \tilde{x}\|_2\right)$ for $L \in \mathbb{R}_+$.

Each of these kernels corresponds to a different feature map, ψ, and in fact, a different Hilbert space, $\mathcal{H}$. Thus, selection of a kernel function will naturally impact the performance of the solution obtained by kernel regression. A key goal of this course is to provide a simple method for selecting effective kernel functions by establishing a connection with neural networks that are known to perform well on a given domain (e.g. convolutional networks on images). We will provide a simple example of this connection at the end of this lecture. For now, we introduce the kernel regression framework by simplifying Eq. (2) given the kernel function notation.

Theorem 2 (Kernel Regression)**.** *Let $\mathcal{H}$ be a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Let $\psi : \mathbb{R}^d \to \mathcal{H}$ and let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a kernel function such that $K(x, \tilde{x}) = \langle \psi(x), \psi(\tilde{x}) \rangle_{\mathcal{H}}$. The minimum $\mathcal{H}$-norm minimizer of the loss:*

$$\mathcal{L}(w) = \frac{1}{2}\sum_{i=1}^{n} (y^{(i)} - \langle w, \psi(x^{(i)}) \rangle_{\mathcal{H}})^2$$

is given by $w^* = \sum_{i=1}^n [y\hat{K}^\dagger]_i \psi(x^{(i)})$ where $\hat{K} \in \mathbb{R}^{n \times n}$ is the positive semi-definite matrix with entries $\hat{K}_{i,j} = K(x^{(i)}, x^{(j)})$. Moreover, the corresponding predictor, $\hat{f} : \mathbb{R}^d \to \mathbb{R}$, is given by

$$\hat{f}(x) = y\hat{K}^\dagger \hat{K}(X, x) \ ;$$

where $\hat{K}(X, x) \in \mathbb{R}^n$ with entries $\hat{K}(X, x)_i = K(x^{(i)}, x)$.

Proof. Following the result of Theorem 1, we need only show that the vector $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n] \in \mathbb{R}^{1 \times n}$ equals $y\hat{K}^\dagger$. In particular, writing Eq. 2 in matrix form, we find:

$$\mathcal{L}(\alpha) = \|y - \alpha\hat{K}\|_2$$

Now minimizing $\mathcal{L}(\alpha)$ is equivalent to solving the linear system $y = \alpha\hat{K}$, which consists of n equations with n variables.[2] In particular the solution is given by $\alpha = y\hat{K}^\dagger$ as was proved in Lecture 2 Theorem 1. Hence we conclude by Theorem 1:

$$w^* = \sum_{i=1}^{n} [y\hat{K}^\dagger]_i \psi(x^{(i)}) \ \ ;$$

[2]The matrix $\hat{K}$ is often referred to as the Gram matrix.

Lastly, recall that our predictor is given by:

$$\hat{f}(x) = \langle w^*, \psi(x) \rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^{n} [y\hat{K}^\dagger]_i \psi(x^{(i)}), \psi(x) \right\rangle_{\mathcal{H}} = y\hat{K}^\dagger \hat{K}(X, x)$$

□

Implementation Remarks. Note that unlike linear regression, Theorem 2 implies kernel regression involves solving linear regression with a square matrix. In particular, this makes kernel regression amenable to fast linear solvers, e.g. using the numpy "solve" function [10]. While traditionally solving a linear system with greater than $50,000$ variables can be time consuming, recent works [4, 5, 6] allow for solving kernel regression with millions and even billions of variables. Importantly, as will be demonstrated in the homework, these methods are far faster than the popular support vector machine (SVM) solvers provided by scikit-learn [8]. Given the importance of such solvers, we provide extensive homework exercises involving running kernel regression with these libraries.

Benefits of Kernel Regression. Note that unlike the complicated loss landscapes of neural networks, finding the minimum norm solution with kernel regression involves solving a convex optimization problem. Hence, optimization becomes conceptually simpler in this setting. Moreover, unlike neural networks, kernel regression offers interpretability of learned solutions in the sense that every prediction on a new sample is just a weighted linear combination of labels for training examples. Hence, akin to the nearest neighbors algorithm, we can understand which training examples were most influential in the prediction for a new sample.

Given the formal development of kernel regression thus far, we now present some concrete examples to make clear how to use the framework in practice.

3 An Empirical Demonstration

We return to the running example of predicting housing prices from square footage from Lecture 2. In particular, we will focus on performing kernel regression using the Gaussian and Laplace kernels. We will importantly understand how altering the kernel bandwidth parameter, i.e. the constant L in the kernel definition, affects the solution given by kernel regression.

Example 2. *Suppose we are again given the square footage of* 10 *houses/apartments in New York City, along with their corresponding price, as shown in Figure 2a. Let* $X = [x^{(1)}, x^{(2)}, \ldots, x^{(10)}]$ *denote the* 10 *training samples and let* $y = [y^{(1)}, y^{(2)}, \ldots, y^{(10)}]$ *denote the* 10 *training labels. We now use kernel regression with kernel* K *being the Laplace or Gaussian kernel to find a nonlinear fit to the data. We outline the steps for kernel regression below.*

Step 1: *Compute the kernel (Gram) matrix* $\hat{K} \in \mathbb{R}^{10 \times 10}$ *where* $\hat{K}_{i,j} = K(x^{(i)}, x^{(j)})$.

Step 2: *Solve for the coefficient vector* α *by solving the linear system of equations* $y = \alpha\hat{K}$.

Step 3: *For any new sample* x, *compute the prediction given by* $\hat{f}(x) = \alpha\hat{K}(X, x)$ *where* $\hat{K}(X, x)_i = K(x^{(i)}, x)$.

In Figure 2, we visualize the kernel regression solution given by Theorem 2 upon varying the choice of kernel (between Gaussian and Laplace) and varying the kernel bandwidth parameter L. *Note that since we are solving kernel regression exactly and since the Gram matrix* $\hat{K}$ *is invertible, all of our predictors interpolate (achieve zero training error) on the data.*

Implementation Suggestions. Note that for small L, e.g. $L = 0.5$, the Laplace kernel provides a reasonable fit to the data. In particular, in the convex hull of the training samples, the predictor achieves low MSE. In practice (and as showcased in the homework), the Laplace kernel is effective and simple to use since the parameter L can be consistently chosen to be a small number, e.g. $L \in \{0.05, 0.1, 0.5\}$. We thus recommend the Laplace kernel as a first kernel to try when establishing a quick, simple baseline with kernel regression.

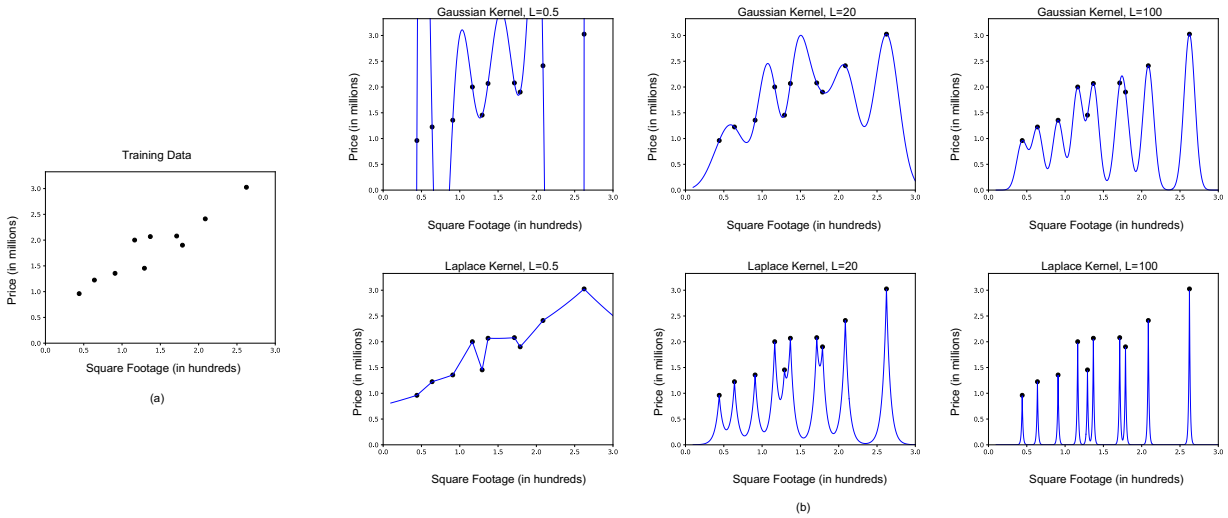


Figure 2: A visualization of solutions given by kernel regression with the Gaussian ($K(x, \tilde{x}) = \exp(-L\|x - \tilde{x}\|_2^2)$) and Laplace kernel ($K(x, \tilde{x}) = \exp(-L\|x - \tilde{x}\|_2)$) for the problem of predicting housing price from square footage. (a) The training data consisting of 10 samples and labels. (b) The solutions given by varying the parameter L in kernel regression with the Gaussian and Laplace kernel. We note that increasing the parameter L leads to a kernel that interpolates the data by spiking at the training samples and outputting near 0 predictions everywhere else. On the other hand, selecting small L ($L = 0.5$) for the Laplace kernel leads to a reasonable interpolating solution with low test MSE.

4 Kernel Regression and Wide Neural Networks

We conclude this lecture with a brief introduction to the connection between training wide neural networks and solving kernel regression. We note that while such connections have been around since the late 1900s [7], recent advancements (in particular, the neural tangent kernel [2]) have renewed interest in this area. Indeed, the remainder of this course will focus on covering key aspects of these recent advances.

For now, we turn to a simplified setting in which we connect nonlinear neural networks where only the last layer is trained to solving kernel regression. In particular, consider the following connection between training the last layer of a 1-hidden layer neural network and solving kernel regression.

Example 3. *For $x \in \mathbb{R}^d$, let $f : \mathbb{R}^d \to \mathbb{R}$ be a 1-hidden layer neural network with elementwise activation $\phi : \mathbb{R} \to \mathbb{R}$ and parameters $a \in \mathbb{R}^{1\times k}, b \in \mathbb{R}^{k\times d}$ such that*

$$f(x) = a\phi(bx) = \frac{1}{\sqrt{k}} \sum_{i=1}^{n} a_i \phi(b_i x) \tag{3}$$

Given training samples $\{x^{(i)}\}_{i=1}^n$ and corresponding labels $\{y^{(i)}\}_{i=1}^n$, let a^* denote the solution given by using gradient descent with initialization $a^{(0)} = \mathbf{0}$ to minimize the loss:

$$\mathcal{L}(a) = \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}))^2 \ ;$$

where the the parameters b are fixed and selected according to $b_{i,j} \overset{i.i.d}{\sim} \mathcal{N}(0, 1)$. The corresponding predictor $\hat{f}(x) = a^*\phi(bx)$ is equivalent to that given by solving kernel regression (e.g. minimizing the loss in Eq. (1)) with the kernel $K(x, \tilde{x}) = \langle \phi(bx), \phi(b\tilde{x}) \rangle_{\ell_2}$.

While this equality holds in the setting where k is finite since $\phi(bx)$ is a fixed feature map, it is remarkably extendable to the setting where the number of hidden units k approaches infinity, and the corresponding

kernel K has a tractable closed form for many nonlinearities ϕ. For example, if ϕ is the popular ReLU activation [1], i.e. $\phi(x) = \max(0, x)$, and $\|x\|_2 = 1, \|\tilde{x}\|_2 = 1$, then as $k \to \infty$, the corresponding kernel is given by:

$$K(x, \tilde{x}) = \frac{1}{\pi}\left(\pi(x^T\tilde{x} - \arccos(x^T\tilde{x})) + \sqrt{1 - (x^T\tilde{x})^2}\right)$$

x^Tx(pi)

In the next lecture, we will introduce such calculations more formally. This will lead to the development of kernels arising as a result of training the last layer of any neural network, i.e. the neural network Gaussian process kernels.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*, volume 1. MIT Press, 2016.

[2] A. Jacot, F. Gabriel, and C. Hongler. Neural Tangent Kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

[3] G. Kimeldorf and G. Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41:495–502, 1970.

[4] S. Ma and M. Belkin. Diving into the shallows: a computational perspective on large-scale shallow learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

[5] S. Ma and M. Belkin. Kernel machines that adapt to GPUs for effective large batch training. In *Conference on Machine Learning and Systems*, 2019.

[6] G. Meanti, L. Carratino, L. Rosasco, and A. Rudi. Kernel methods through the roof: handling billions of points efficiently. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020.

[7] R. Neal. *Bayesian Learning for Neural Networks*, volume 1. Springer, 1996.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research British Machine Vision Association*, 12:2825–2830, 2011.

[9] W. Rudin. Fourier analysis on groups. 1962.

[10] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.