CS 181 Spring 2022 Section 2 Notes: Nonparametric Regression, Linear Regression, MLE, Model Selection

Ty Geri, Luke Bailey

1 KNN and Kernelized Regression

1.1 K-Nearest Neighbors (KNN)

1. KNN is considered a form of non-parametric regression. Intuitively, for a fixed value of K=k, there are no parameters that we have to tune (i.e., no weights that we have to learn). More rigorously, non-parametric means that we do not make any assumptions about the parameters / characteristics of our data (in context, this means that we do not assume an underlying probability distribution).

2. Rundown of the KNN Algorithm:

   (a) Let $\mathbf{x}^*$ be the point that we would like to make a prediction about. Let's find the k nearest points $\{x_1, \dots x_k\}$ to $\mathbf{x}^*$, based on some predetermined distance function.

   (b) Denote the true y values of these k points as $\{y_1, \dots y_k\}$.

   (c) Output our prediction $\hat{y}^*$ for our point of interest $\mathbf{x}^*$:

   $$\hat{y}^* = \frac{\sum_{i=1}^{k} y_k}{k}$$

   Remark: the little "hat" in $\hat{y}^*$ is just notation telling us that this is our *prediction*, rather than the true y value.

1.2 Kernelized Regression

Note: You might come across this concept under the name kernel-weighted average regression.

1. Kernelized Regression is considered to be a smoother, more general extension of KNN.

2. Let $k(\mathbf{x}^*, x_n)$ be our "kernel function." In Kernelized Regression, we want to take a *weighted average* of all the points in our training data when outputting our prediction for an unknown point. Intuitively, we want to weigh points that are "closer" to our unknown point of interest *more heavily* than points that are "farther" away. As such, our kernel function $k(\mathbf{x}^*, \mathbf{x_n})$ should be *larger* for a point $\mathbf{x_n}$ closer to our point of interest $\mathbf{x}^*$ than a point $\mathbf{x_n}$ farther away.

3. Importantly, the following should always hold:

   $$\arg\max_x k(\mathbf{x}^*, x) = \mathbf{x}^*$$

   Remark: What the above statement means is that "the value of x that results in the largest value of $k(\mathbf{x}^*, x)$ should be $\mathbf{x}^*$ itself".

4. Rundown of the Kernelized Algorithm:

   (a) Let $\{x_1, \ldots x_N\}$ (and their corresponding y values) be *all* of the N points comprising our training data set.

   (b) Let $\mathbf{x}^*$ be our point of interest that we want to make a prediction for. We make our prediction as follows:

$$\hat{y}^* = \frac{\sum_{i=1}^{N} k(\mathbf{x}^*, x_i) \cdot y_i}{\sum_{j=1}^{N} k(\mathbf{x}^*, x_j)}$$

   Remark: notice how we include *all* points of the training data in our calculation?

   Remark: we have to include the denominator term in order to normalize the sum of our weights to equal 1.

2 Probabilistic Regression

Last time we derived an analytical solution for least squares linear regression. When doing this we picked the following loss function for our model and found the weights that minimized it:

$$w = \frac{1}{2} \sum_{n=1}^{N} (y_n - w^n)^2 \tag{1}$$

We chose this loss function **because it seemed sensible**. There are many other loss functions that also seem sensible, for example:

$$w = \frac{1}{2} \sum_{n=1}^{N} |y_n - w^n| \tag{2}$$

However, using this loss function would result in different learned weights for our regression model given the same training data. It can thus be difficult to see which loss function we should use. In probabilistic regression, we make an assumption about how the data was generated, and from this use statistical methods to derive a value for the weights. We will see that doing this will connects back to our least squares loss function.

2.1 Story about data generation

Suppose we have a dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$. where $x_n \in \mathbb{R}^D$. We will assume that the target variables y are generated from the data x as follows:

$$y = w^T x + \epsilon, \;\; \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{3}$$

Where ϵ is the noise. We will treat σ as some known value that we are given. From the above, we have that each $y|x, w$ is a random variable with the following distribution:

$$y|x, w \sim \mathcal{N}(w^T x, \sigma^2) \tag{4}$$

2.2 Likelihood

For our model, we will select some value for w. We denote the joint likelihood of the model as:

$$p(y_1, \ldots, y_N | x_1, \ldots, x_N, w) = \prod_{i=1}^{N} p(y_i | x_i, w) \tag{5}$$

We have the above equality as we assume that the data is i.i.d, and each y_i only depends on the corresponding x_i.

Intuition: What is the likelihood? Looking at the equation, it tells us for some choice of w what the probability of the dataset being generated using that w is. From this, it would make sense that we want to find a w that **maximises the likelihood** of the data. We call this value the maximum likelihood estimate for w, denoted by w^{MLE}. We can encode this objective as follows:

$$\begin{aligned}
w^{MLE} &= \arg\max_w \; p(y_1, \ldots, y_N | x_1, \ldots, x_N, w) && (6)\\
&= \arg\max_w \prod_{i=1}^{N} p(y_i | x_i, w) && (7)\\
&= \arg\max_w \log\left(\prod_{i=1}^{N} p(y_i | x_i, w)\right) && (8)\\
&= \arg\max_w \sum_{i=1}^{N} \log(p(y_i | x_i, w)) && (9)
\end{aligned}$$

From equation (7) to (8) we note that maximising a function is the same as maximising the log of that function (because log is a monotonically increasing function). From steps (8) to (9) we simply use log laws. Specifically $\log(ab) = \log(a) + \log(b)$, which allows us to convert the product into a sum. This is known as the **log likelihood**.

Aside: Why do we optimize the log likelihood and not the likelihood? To find the maximising w we will take the derivative of the function we want to optimize, set the derivative to 0 and then solve for w. Taking the derivative of the likelihood is very difficult because it is a product of many terms, so we would have to use the product rule over and over again. Taking the derivative of the log likelihood is much easier however as it is a sum, so we can just differentiate each term seperately. Additionally, on computers working with the log likelihood is more numerically stable, as the likelihood can end up being very small (it is the product of N terms, all of which are between 0 and 1), so small that the precision of floating point numbers starts affecting calculations.

2.3 Finding W^{MLE} for linear regression

In general, to find the maximum likelihood estimator (MLE), we take the following steps:

1. Find the log likelihood of the data.
2. Differentiate with respect to the parameter you want to find an MLE for (in our case w).
3. Set the derivative equal to 0 and solve for you parameter (more precisely, find the stationary point).

Returning back to our example, we have the log likelihood $\ell(w)$ as:

$$\ell(w) = \log\left[\prod_{n=1}^{N} p(y_n|x_n, w)\right] = \sum_{n=1}^{N} \log p(y_n|x_n, w) \tag{10}$$

$$\tag{11}$$

We now plug in the Gaussian pdf, remembering that we have $y_n|x_m, w \sim \mathcal{N}(w^T x_n, \sigma^2)$:

$$\ell(w) = \log\left[\prod_{n=1}^{N} p(y_n|x_n, w)\right] = \sum_{n=1}^{N} \log p(y_n|x_n, w) \tag{12}$$

$$= \sum_{n=1}^{N} \log\left[\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) \exp\left\{-\frac{(y_n - w^T x_n)^2}{2\sigma^2}\right\}\right] \tag{13}$$

$$= \sum_{n=1}^{N} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\right] - \sum_{n=1}^{N} \frac{(y_n - w^T x_n)^2}{2\sigma^2}. \tag{14}$$

To maximize the log-likelihood $\ell(w)$, we compute the gradient of $\ell(w)$ with respect to w, set it equal to 0 and solve for w.

$$\nabla\ell(w) = \nabla\left(\sum_{n=1}^{N} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\right] - \sum_{n=1}^{N} \frac{(y_n - w^T x_n)^2}{2\sigma^2}\right) \tag{15}$$

$$= \nabla\sum_{n=1}^{N} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\right] - \nabla\sum_{n=1}^{N} \frac{(y_n - w^T x_n)^2}{2\sigma^2} \tag{16}$$

$$= -\frac{1}{2\sigma^2}\nabla\left(\sum_{n=1}^{N}(y_n - w^T x_n)^2\right) \tag{17}$$

$$= -\frac{N}{2\sigma^2}\nabla\left(\frac{1}{N}\sum_{n=1}^{N}(y_n - w^T x_n)^2\right) \tag{18}$$

$$\tag{19}$$

We see that the gradient of the log-likelihood is the same (within a multiplicative factor) as the gradient of the mean squared error loss function shown in equation (1)! Thus minimizing

(1) is the same as maximizing the log-likelihood - we flip from minimizing to maximizing as the multiplicative factor between the two is negative. This means they will have the same stationary points. We will not go through the algebra now to find these stationary points as we already did it for mean squared error loss last week.

Concept check: What does this tell us? It means that if we assume our data is generated in the way we outlines at the start of section 2.1, then using mean squared error loss is the same as finding the MLE for w. This gives us evidence that using that loss function might be good. What if we thought our data was generated a different way however? For example if we were measuring a physical process and we knew that the noise of our measuring instrument was not normal? In this case the MLE may no longer correspond to minimizing least squares loss.

2.4 Properties of MLE

Maximum likelihood estimators w^{MLE} for parameters w, have the following properties:

1. **Consistent:** As you sample more and more data, w^{MLE} will converge in probability to w.
2. **Asymptotically unbiased:** In the limit as you have more and more data, w^{MLE} will be unbiased. We of course do not have infinite data however, so our MLE estimates may be biased.
3. **Asymptotically minimal variance:** In the limit as you have more and more data, w^{MLE} will be the lowest variance estimator for w (of all asymptotically unbiased estimators).

The above properties make the MLE a useful and desirable estimator.

2.5 Exercise: MLE for Gaussian Data

We are given a data set $(x_1, \ldots, x_n)$ where each observation is drawn independently from a multivariate Gaussian distribution:

$$\mathcal{N}(x|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{|(2\pi)\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(x-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})\right) \tag{20}$$

where $\boldsymbol{\mu}$ is a m-dimensional mean vector, $\boldsymbol{\Sigma}$ is a m by m covariance matrix, and $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$.

1. Find the log likelihood $\log p(x_1, \ldots, x_n|\mu, \Sigma)$.
2. (Harder) Find the maximum likelihood value of the mean, $\boldsymbol{\mu}_{MLE}$, assume that Σ is given (matrix cookbook ch 2.4 eqn 78)

Solution: We find the MLE by maximizing the log likelihood:

$$\begin{aligned} l(\boldsymbol{\mu}, \boldsymbol{\Sigma}; x) &= \log\left(\prod_{i=1}^{n} \mathcal{N}(x_i|\boldsymbol{\mu}, \boldsymbol{\Sigma})\right) = \sum_{i=1}^{n} \log(\mathcal{N}(x_i|\boldsymbol{\mu}, \boldsymbol{\Sigma})) \\ &= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(|\boldsymbol{\Sigma}|) - \frac{1}{2}\sum_{i=1}^{n}(x_i-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(x_i-\boldsymbol{\mu}) \end{aligned}$$

Taking the derivative (matrix cookbook ch 2.4 eqn 78) with respect to $\boldsymbol{\mu}$ and setting it equal to 0, we get

$$0 = \frac{\partial l}{\partial \boldsymbol{\mu}} = \sum_{i=1}^{n} \boldsymbol{\Sigma}^{-1}(x_i - \boldsymbol{\mu})$$

and solving gives us that

$$\boldsymbol{\mu}_{MLE} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

3 Bias-Variance Trade-Off

3.1 Motivation

As model makers, before we train a model we decide on a model class to use. For example we could use a simple linear regression, or a linear regression with a fancy basis function applied. We want to choose a model class that is expressive enough to learn the underlying function that generated the data (avoid underfitting), but not so expressive that it learns the training set perfectly and doesn't generalize (avoid overfitting). By expressive we sort of mean how many different functions can the model learn. Note that once you have chosen a model class, f, the final learned model f_w is random, dependent on what our training set $\mathcal{D}$ is. We consider two different cased for our model class f:

1. f is too expressive. In this case, f_w will overfit the training data $\mathcal{D}$. This means that if we randomly draw a new dataset $\mathcal{D}'$, and train a new model of f on this data, this new model, $f_{w'}$, will overfit the new dataset $\mathcal{D}'$. Consequently, f_w and $f_{w'}$ will be very different models. Put more formally, for a random testing point (x, y), $f_w(x)$ and $f_{w'}(x)$ will be very different. Our model class f is **very sensitive** to the choice of training data. This means the model has **high variance**.

   Despite this, if we trained many versions of f on many randomly drawn datasets $\mathcal{D}$, and averaged the value $f_w(x)$ for each, getting some prediction $\hat{y}$, we would expect $\hat{y} \approx y$. That is in expectation, $f_w(x) \approx y$. The means the model has **low bias**.

2. f is not expressive enough. In this case, f_w will underfit the training data $\mathcal{D}$. This means if we randomly draw a new dataset $\mathcal{D}'$, and train a new model $f_{w'}$, it will also underfit the new dataset $\mathcal{D}'$, resulting in $f_w(x) \approx f_{w'}(x)$ for some new random testing point (x, y). So the model class f is **not sensitive** to the choice of training data. Thus f is **low variance**.

   Despite this, if we trained many versions of f on many randomly drawn datasets $\mathcal{D}$, and averaged the value $f_w(x)$ for each, getting some prediction $\hat{y}$, we would expect $\hat{y} \not\approx y$. That is in expectation, $f_w(x) \not\approx y$. A random model of f will always underfit the data and so we don't expect its predictions to be good. Thus f has **high bias**.

Ideally we want a model that is low variance and low bias. From the above there seems to be a tradeoff between the two however. As we make our models more expressive, the bias will go down but the variance will go up. One of the most important things you will learn in this class is how important choosing you model class is, and that is rooted in the **bias variance tradeoff** we have outlined here. We now derive the existence of the tradeoff theoretically to get a better understanding of what is going on.

3.2 Bias-Variance Decomposition

Bias-variance decomposition is a way of understanding how different sources of error (bias and variance) can affect the final performance of a model. A tradeoff between bias and variance is often made when selecting models to use. Let's work step-by-step to decompose the MSE / generalization error into a more interpretable form:

$$\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[(y - f_{\mathbf{w}}(\mathbf{x}))^2\right] \tag{21}$$

This is the MSE of a trained model $f_{\mathbf{w}}$ using a random drawn dataset $\mathcal{D}$ for a random single test point (x, y). We start by adding and subtracting $\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]$:

$$\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[(y - f_{\mathbf{w}}(\mathbf{x}))^2\right] = \mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[((y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]) + (\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})))^2\right] \tag{22}$$

When we expand the square we get:

$$= \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right])^2\right]}_{\text{Term 1}} \tag{23}$$

$$+ \underbrace{2\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right])\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)\right]}_{\text{Term 2}} \tag{24}$$

$$+ \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)^2\right]}_{\text{Term 3}} \tag{25}$$

Before we move on let's do a quick review of Adam's Law which shows that:

$$\mathbb{E}_{(\mathbf{x},y),\mathcal{D}} = \mathbb{E}_{(\mathbf{x},y)}\mathbb{E}_{\mathcal{D}|(\mathbf{x},y)} \tag{26}$$

However, since the test data points are sampled independently from the training data:

$$\mathbb{E}_{(\mathbf{x},y),\mathcal{D}} = \mathbb{E}_{(\mathbf{x},y)}\mathbb{E}_{\mathcal{D}} \tag{27}$$

Using this and the fact that $\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]$ and $f_{\mathbf{w}}(\mathbf{x})$ don't depend on $y|x$ we show that Term 2 (23) is equal to 0:

$$\underbrace{2\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right])\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)\right]}_{\text{Term 2}} \tag{28}$$

$$= 2\mathbb{E}_{(\mathbf{x},y)}\mathbb{E}_{\mathcal{D}}\left[(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right])\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)\right] \tag{29}$$

Note that $(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathrm{w}}(\mathbf{x})\right])$ is constant with respect to the dataset $\mathcal{D}$ so we can pull it out as a constant:

$$= 2\mathbb{E}_{(\mathbf{x},y)}\left[(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right])\,\mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right]\right] \tag{30}$$

$$= 2\mathbb{E}_{(\mathbf{x},y)}\left[(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right])\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)\right] \tag{31}$$

$$= 2\mathbb{E}_{(\mathbf{x},y)}\left[(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]) \cdot 0\right] \tag{32}$$

$$= 0 \tag{33}$$

Now let's re-write the decomposition without Term 2:

$$\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - f_{\mathbf{w}}(\mathbf{x})\right)^2\right] \tag{34}$$

$$= \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right]}_{\text{Term 1}} \tag{35}$$

$$+ \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)^2\right]}_{\text{Term 3}} \tag{36}$$

Now we can decompose Term 1 (32) to make it more interpretable. We do the same trick of adding and subtracting a $\mathbb{E}_{y|\mathbf{x}}[y]$ term:

$$\underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right]}_{\text{Term 1}} = \mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right) + \left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)\right)^2\right] \tag{37}$$

When we expand the square we get:

$$\underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right]}_{\text{Term 1}} \tag{38}$$

$$= \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)^2\right]}_{\text{Term 1}A}. \tag{39}$$

$$+ \underbrace{2\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)\left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)\right]}_{\text{Term 1}B} \tag{40}$$

$$+ \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right]}_{\text{Term 1}C} \tag{41}$$

But note that, as in Term 2 (23), we can apply Adam's Law and independence again:

$$\mathbb{E}_{(\mathbf{x},y),\mathcal{D}} = \mathbb{E}_{(\mathbf{x},y)}\mathbb{E}_{\mathcal{D}} = \mathbb{E}_{\mathcal{D}}\mathbb{E}_{(\mathbf{x},y)} = \mathbb{E}_{\mathcal{D}}\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{y}|\mathbf{x}} \tag{42}$$

This will allow us to show that Term 1B is also equal to 0:

$$\underbrace{2\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)\left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)\right]}_{\text{Term 1}B} \tag{43}$$

$$= 2\mathbb{E}_{\mathcal{D}}\mathbb{E}_{\mathbf{x}}\mathbb{E}_{y|\mathbf{x}}\left[\left(y - \mathbb{E}_{(y|\mathbf{x})}[y]\right)\left(\mathbb{E}_{(y|\mathbf{x})}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)\right] \tag{44}$$

Note that $\left(\mathbb{E}_{(y|\mathbf{x})}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)$ is constant with respect to $y \mid x$ so when we can pull it out as a constant in front of $\mathbb{E}_{y|\mathbf{x}}\left(y - \mathbb{E}_{(y|\mathbf{x})}[y]\right)$:

$$= 2\mathbb{E}_{\mathcal{D}}\mathbb{E}_{\mathbf{x}}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)\mathbb{E}_{y|\mathbf{x}}\left[y - \mathbb{E}_{y|\mathbf{x}}[y]\right]\right] \tag{45}$$

$$= 2\mathbb{E}_{\mathcal{D}}\mathbb{E}_{\mathbf{x}}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)\left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{y|\mathbf{x}}[y]\right)\right] \tag{46}$$

$$= 2\mathbb{E}_{\mathcal{D}}\mathbb{E}_{\mathbf{x}}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)\cdot 0\right] \tag{47}$$

$$= 2\mathbb{E}_{\mathcal{D}}\mathbb{E}_{\mathrm{x}}[0] \tag{48}$$

$$= 0 \tag{49}$$

Therefore, Term 1 can be re-written as:

$$\underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right]}_{\text{Term 1}} = \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)^2\right]}_{\text{Term 1}A} \tag{50}$$

$$+ \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right]}_{\text{Term 1}C} \tag{51}$$

Now we can re-write our MSE / generalization error with our more interpretable Term 1:

$$\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - f_{\mathbf{w}}(\mathbf{x})\right)^2\right] = \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)^2\right]}_{\text{Term 1}A}. \tag{52}$$

$$+ \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right]}_{\text{Term 1}C} \tag{53}$$

$$+ \underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)^2\right]}_{\text{Term 3}} \tag{54}$$

3.3 Interpretation

Now we can finally interpret our MSE in terms of the Bias-Variance trade-off.

1. **Term 1A**: Since $(\mathbf{x}, y)$ is independent of $\mathcal{D}$ (test data and training are independent), we can re-write this as:

$$\underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)^2\right]}_{\text{Term 1}A} = \mathbb{E}_{(\mathbf{x},y)}\left[\left(\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)^2\right)\right] \tag{55}$$

   Applying Adam's Law we can derive:

$$\mathbb{E}_{(\mathbf{x},y)}\left[\left(\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)^2\right)\right] = \mathbb{E}_{\mathbf{x}}\mathbb{E}_{y|\mathbf{x}}\left[\left(\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)^2\right)\right] \tag{56}$$

   Recall that $\mathbb{E}_{y|\mathbf{x}}\left[\left(\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)^2\right)\right]$ is $\mathrm{Var}[y \mid \mathbf{x}] = \sigma^2$ and therefore **Term 1A** is:

$$\mathbb{E}_{\mathbf{x}}[\sigma^2] = \sigma^2 \tag{57}$$

   which is the *Observation Noise*.

2. **Term 1C**: Since $(\mathbf{x}, y)$ and $\mathbb{E}_{\mathrm{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]$ are independent of $\mathcal{D}$ we can re-write Term 1C as:

$$\underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right]}_{\text{Term } 1C} = \mathbb{E}_{(\mathbf{x},y)}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y \mid \mathbf{x}] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right] \tag{58}$$

Applying Adam's Law we can derive:

$$\mathbb{E}_{(\mathbf{x},y)}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y \mid \mathbf{x}] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right] = \mathbb{E}_{\mathbf{x}}\mathbb{E}_{y|\mathbf{x}}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y \mid \mathbf{x}] - \mathbb{E}_{\mathrm{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right] \tag{59}$$

$$= \mathbb{E}_{\mathbf{x}}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y \mid \mathbf{x}] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right] \tag{60}$$

where $\mathbb{E}_{y|\mathbf{x}}[y \mid \mathbf{x}]$ is the average true y value and $\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right])$ is the average predicted y value over the training data. This is the *Bias* term.

3. **Term 3**: Since the term is independent of y we can simplify as such:

$$\underbrace{\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)^2\right]}_{\text{Term } 3} = \mathbb{E}_{\mathbf{x},\mathcal{D}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)^2\right] \tag{61}$$

$$= \mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathcal{D}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)^2\right] \tag{62}$$

where $\mathbb{E}_{\mathcal{D}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)^2\right] = \mathrm{Var}\left[f_{\mathbf{w}}(\mathbf{x})\right]$ and therefore **Term 3** is:

$$\mathbb{E}_{\mathbf{x}}\left[\mathrm{Var}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right] \tag{63}$$

which is the *variance* of the model.

3.4 Conclusion

Summing this all together we can interpret the MSE / generalization error as:

$$\mathbb{E}_{(\mathbf{x},y),\mathcal{D}}\left[\left(y - f_{\mathbf{w}}(\mathbf{x})\right)^2\right] = \underbrace{\mathbb{E}_{y|\mathbf{x}}\left[\left(\left(y - \mathbb{E}_{y|\mathbf{x}}[y]\right)^2\right)\right]}_{\text{noise}} \tag{64}$$

$$+ \underbrace{\mathbb{E}_{\mathbf{x}}\left[\left(\mathbb{E}_{y|\mathbf{x}}[y \mid \mathbf{x}] - \mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right]\right)^2\right]}_{\text{bias}^2} \tag{65}$$

$$+ \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathbf{w}}(\mathbf{x})\right] - f_{\mathbf{w}}(\mathbf{x})\right)^2\right]}_{\text{variance}} \tag{66}$$

$$= \mathbb{E}_{\mathbf{x}}\left[\text{noise}(\mathbf{x}) + \text{bias}^2(f(\mathbf{x})) + \text{variance}\left(f_{\mathbf{w}}(\mathbf{x})\right)\right] \tag{67}$$

4 Validation Sets and Ensemble Methods

A *validation set* contains data that are separate from our training set used to fit the regression. Here is a sample process:

1. Separate our full dataset into a training set and validation set (say in a 90/10 split).
2. Train your models with different parameters on the training set. Each time, check the performance on the validation set.
3. This gives you an optimal value for the parameter.

4.1 Cross Validation

Cross validation is a more sophisticated technique for obtaining validation losses.

1. In k-fold cross validation, we split our data into k equal chunks.
2. For each chunk, we set it to be the validation set and use the rest of the $k-1$ chunks together as the training data to fit our model.
3. Then, we obtain a validation loss on our current chunk.
4. Averaging loss over the k chunks gives us a final validation loss.

Now, we have an improved way of computing validation losses by averaging. This reduces the variance in the resulting validation loss - since we've used every data point in doing so!

4.2 Ensemble Methods

Ensemble methods take advantage of multiple models to obtain better predictive accuracy than with a single model alone. The two most common types are bagging and boosting.

4.2.1 Bootstrap aggregating (Bagging)

- In bagging, we fit each individual model on a random sample of the training set.
- To predict data in the test set, we either use an average of the predictions from the individual models (for regression) or take the majority vote (for classification).
- This tends to lower variance without changing bias, since it's an average of models!
- **Example:** Random forest, which is an average of predictions from decision trees!

4.2.2 Boosting

- In boosting, we train the individual models sequentially. After training the i^{th} model on a sample of the training set, we train the $(i+1)^{th}$ model on a new sample based on the performance of the i^{th} model.
- Thus, examples classified incorrectly in the previous step receive higher weights in the new sample, encouraging the new model to focus on those examples.
- During testing, we take a weighted average or weighted majority vote of the models' predictions based on their respective training accuracies on their reweighted training data (i.e. higher models have larger weights).
- **Example:** The Adaboost algorithm is a common example.

5 Regularization

5.1 Linear Regression

Suppose we have data $\{(x_i, y_i)\}_{i=1}^N$, with $x_i, y_i \in \mathbb{R}$, and we want to fit polynomial basis functions:

$$\mathbf{x}^\top = [\phi_1(x) = 1, \phi_2(x) = x, \ldots, \phi_{d+1}(x) = x^d]$$

$$f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$$

That is, we fit a degree d polynomial. With a small dataset and too high of a d, we get overfitting. Obviously, this will generalize poorly to new data points. How can we solve this problem?

5.2 Penalized Loss Function

Recall that the standard linear regression problem, known as uses the following loss function (which is just the mean squared error):

$$\mathcal{L}_{OLS}(D) = MSE = \sum_{i=1}^{N} (y_i - f(x_i; \mathbf{w}))^2$$

Regularization refers to the general practice of modifying the model-fitting process to avoid overfitting. Linear models are typically regularized by adding a *penalization term* to the loss function. The penalization term is simply any function R of the weights $\mathbf{w}$ scaled by a penalization factor λ. The loss then becomes:

$$\mathcal{L}_{reg}(D) = \sum_{i=1}^{N} (y_i - f(x_i; \mathbf{w}))^2 + \lambda R(\mathbf{w})$$

There are some common choices for $R(\mathbf{w})$ that will be discussed. They frequently leverage the idea of a vector norm, where $||\mathbf{w}||_p$ represents the L_p-norm of the vector $\mathbf{w}$ for $p \geq 1$:

$$||\mathbf{w}||_p = \left(\sum_d |\mathbf{w}_d|^p \right)^{1/p}$$

5.3 LASSO Regression

One common choice for a penalization term is simply $R(\mathbf{w}) = ||\mathbf{w}||$. This is just the L_1-norm of the weights vector, which quite naively means that the penalization term here is just the sum of the magnitudes of all the weights for the model. This form of regularized regression is known as *LASSO (Least Absolute Shrinkage and Selection Operator)* regression. The full modified loss is then:

$$\mathcal{L}_{LASSO}(D) = \sum_{i=1}^{N} (y_i - f(x_i; \mathbf{w}))^2 + \lambda ||\mathbf{w}||$$

There are some notable properties of LASSO regression. One main disadvantage is that it does not have a closed-form solution, meaning that it cannot be analytically solved. Therefore, it needs to be numerically solved through an iterative process, which can be much slower.

Concept Question: Why do you think LASSO has no closed-form solution? Try to solve for it using the same process as for the OLS solution; what goes wrong?

However, it does have the benefit, as the name suggests, that it is good for *variable selection*, meaning that coefficients for some variables (ones that have low predictive power) might be "shrunk" directly to zero.

5.4 Ridge Regression

Another solution to overfitting linear regression is through ridge regression, which minimizes a modified least squares loss function:

$$\mathcal{L}(D) = \sum_{i=1}^{N}(y_i - f(x_i; \mathbf{w}))^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

Ridge regression is used to *regularize* a model, making it simpler and allowing it to generalize better to new data. Indeed, the extra term penalizes overly large weights in $\mathbf{w}$, leading to smaller coefficients for a "flatter" polynomial:

Unlike LASSO, ridge regression has a closed form solution, which makes the solution much more computationally efficient. While it does not shrink coefficients to zero, it has other intuitive properties, such as connection to a Normal prior. The analytical solution is:

$$\mathbf{w}_{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

The above expression can be compared to the OLS solution. The only additional term is $\lambda\mathbf{I}$, which looks like a "ridge" of λ values (hence the name ridge regression). This also helps avoid problems with singular data.