Chapter 2

Is it better to have $P_R = P_S$?

While great effort has been spent in the literature trying to match the training and test distributions, a thorough analysis of the need for matching has not been carried out. In particular, the first fundamental question we asked in Chapter 1 has not been answered: is it better to have training and test distributions matched, in terms of out-of-sample performance? As we will show shortly, the main contribution in this chapter is to show that mismatched distributions can in fact outperform matched distributions in the supervised learning setting, *regardless* of the specific target function. We first published the results of this chapter in [36].

This statement is not very surprising if we are under the active learning paradigm. Under such paradigm, training data is selected sequentially or in batches, making use of feedback obtained from the target function. The goal is to select the least amount of training data that will lead to the best performance. Therefore, the choice of such data may be tilted towards regions that best pin down the target function further, irrespective of the test distribution. Some methods belonging to the active learning paradigm exploit this idea by finding a 'design' distribution, from which the training data should be sampled. The idea is that training the algorithm with data sampled from the design distribution would result in better performance. Examples of these techniques are found in [70], [43], [63], [66], [62], and [58], among others. Hence, it is clear that the active learning paradigm makes use of unmatched distributions to improve performance.

In the supervised learning paradigm, however, the location of the training data is chosen without any feedback from the target function. Therefore, it is more surprising in this case that a data distribution that is mismatched to the test distribution would perform better. Recognizing that the system may perform better under a scenario of mismatched distributions can influence the need for, and the extent of, matching techniques, as well as the quantitative objective of matching algorithms.

In our analysis, we show that a mismatched distribution can be better than a matched distribution in two different directions in the supervised learning paradigm:

- For a given training distribution P_R, the best test distribution P_S can be different from P_R.
- For a given test distribution P_S, the best training distribution P_R can be different from P_S.

The justifications for these two directions, as well as their implications, are quite different. In a practical setting, the test distribution is usually fixed, so the second direction reflects the practical learning problem about what to do with the training data if it is drawn from a different distribution than that of the test environment. One of the ramifications of this direction is the new notion of a *dual distribution*. This is a training distribution P_R that is optimal to use when the test distribution is P_S, regardless of the specific target function. A dual distribution serves as a new objective for matching algorithms. Instead of matching the training distribution to the test distribution, it is matched to a dual of the test distribution, for optimal performance.

We cover both classification and regression settings in the sections that follow. The classification setting is analyzed through empirical results obtained via Monte Carlo simulations. We then present both empirical and analytic results in the regression setting.

2.1 Empirical results in the classification setting

Consider the learning scenario where the data set R used for training by the learning algorithm is drawn from probability distribution P_R, while the data set S that the algorithm will be tested on is drawn from distribution P_S. We show here that the performance of the learning algorithm in terms of the out-of-sample error can be better when $P_S \neq P_R$, averaging over target functions and data set realizations. The empirical evidence, which is statistically significant, is based on an elaborate Monte Carlo simulation that involves various target functions and probability distributions. The details of that simulation follow, and the results are illustrated in Figures 2.1 and 2.3.

We consider the input space $\mathcal{X} = [-1, 1]$. There is no loss of generality by limiting our domain as in any practical situation, the data has a finite domain and can be rescaled to the desired interval. We pick a one-dimensional space to have a better understanding in this simpler case, before generalizing to multiple dimensions. We run the learning algorithm for different target functions and different training and test distributions. We then average the out-of-sample error over a large number of data sets generated by those distributions and over target functions. Finally we compare the results for matched and mismatched distributions.

Distributions. We use 31 different probability distributions to generate R and S including a uniform distribution $U(-1, 1)$, ten truncated Gaussian distributions $\mathcal{N}^*(0, \sigma^2)$ where σ is increased in steps of 0.3, ten truncated exponential distributions $Exp^*(\tau)$ where τ is increased also in steps of 0.3, and ten truncated mixture of Gaussian distributions, such that $MG^*(\sigma) = \frac{1}{2}\left(\mathcal{N}^*(-0.5, \sigma^2) + \mathcal{N}^*(0.5, \sigma^2)\right)$,

with σ increased in steps of 0.25. By truncating the distributions we mean that we zero-out the probability distributions outside $\mathcal{X}$ and renormalize the densities accordingly. That is, if X has a truncated Gaussian distribution such that $X \sim \mathcal{N}^*(0, \sigma^2)$ and $\tilde{X}$ has a Gaussian distribution with $\tilde{X} \sim \mathcal{N}(0, \sigma^2)$, then

$$P(X \leq x) = \begin{cases} 0 & x \leq -1 \\ \frac{1}{Z} P(\tilde{X} \leq x) & -1 \leq x \leq 1 \\ 1 & x \geq 1 \end{cases} \tag{2.1}$$

where $Z = P(-1 \leq \tilde{X} \leq 1)$. Similarly, this applies for the truncated Exponential and Mixture of Gaussian distributions.

Data Sets. For each pair of probability distributions, we carry out the simulation generating 1,000 different target functions, running the learning algorithm, comparing the out-of-sample performance, and then averaging over 100 different data set realizations. That is, each point in Figures 2.1 and 2.3 is an average over 100,000 runs with the same pair of distributions but with different combinations of target functions and training and test sets. The sizes of the data sets are $N_R = 100$ and 300, and $N_S = 10,000$, where N_R and N_S are the number of points in the training and test sets R and S.

Target Functions. The target functions $f : [-1, 1] \to [-1, 1]$ were generated by taking the sign of a polynomial in the desired interval. The polynomials were formed by choosing at random one to five roots in the interval [-1,1]. This choice of target functions allows the decision boundaries to vary both in number and location in each realization. Hence, the results presented do not depend on a particular target function, so that the distributions cannot favor the regions around the boundaries, as these are changing in each realization.

Learning model The learning algorithm minimized a squared loss function. For the hypothesis set $\mathcal{H}$ we used linear functions of a non-linear transformation of the input space. The non-linear transformation used powers of the input variable up to the number of roots of the polynomial that describes the target function, plus a sinusoidal feature, which allows the model to learn a function that is close to, but not identical to, the target. That is, for every $h \in \mathcal{H}$

$$h(x; \theta) = \theta^T \phi_M(x), \tag{2.2}$$

with $\phi_M : \mathcal{X} \to \mathbb{R}^M$, and $\theta \in \mathbb{R}^M$, where

$$\phi_M(x) = [1 \quad x \quad x^2 \cdots \quad x^{M-2} \quad \sin(\pi x)]^T. \tag{2.3}$$

Out-of-sample error. The expected out-of-sample error E_{out} in this classification task is estimated using the test set generated according to each of the P_S with $N_S = 10,000$. The error at a

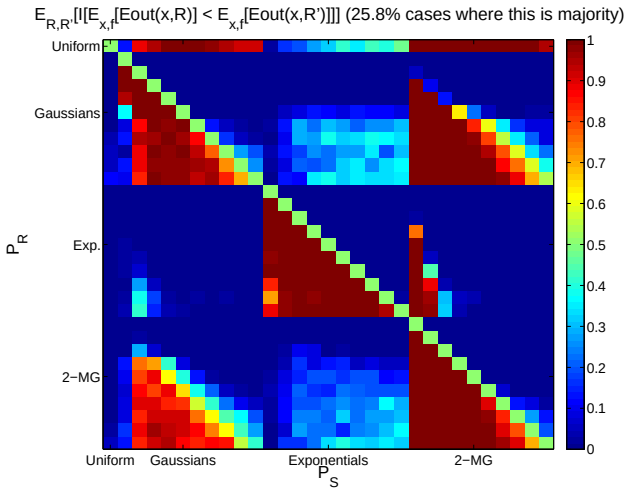

Figure 2.1: Summary of Monte Carlo Simulation. Plot indicates, for each combination of probability distributions $\mathbb{E}_{R\sim P_R, R'\sim P_S}[I[\mathbb{E}_{x\sim P_S, f}[E_{\text{out}}(x, R, f)] < \mathbb{E}_{f,x\sim P_S}[E_{\text{out}}(x, R', f)]]]$.

point $x \in \mathcal{X}$ depends not only on the point itself, but also on the training data set R used for learning which in turn affects the learned hypothesis $g \in \mathcal{H}$, and also depends on the target function f. We compute E_{out} using the misclassification 0-1 loss, that is

$$\mathbb{E}_{x,R}[E_{\text{out}}(x, R, f)] = \mathbb{E}_{x,R}[I[f(x) \neq g(x)]], \tag{2.4}$$

where $I[a]$ denotes the indicator function of expression a, $x \sim P_S$, and R is generated according to P_R.

2.1.1 Fixing the training distribution

Figure 2.1 summarizes the result of the simulation to answer the question in the first direction: for a given training distribution, is the best test distribution different? Each entry in the matrix corresponds to a pair of distributions P_R and P_S. We fix P_R, and evaluate the percentage of runs where using $P_S \neq P_R$ yields better out-of-sample performance than if $P_S = P_R$. That is, each entry corresponds to

$$\mathbb{E}_{R\sim P_R, R'\sim P_S}[I[\mathbb{E}_{f,x\sim P_S}[E_{\text{out}}(x, R, f)] < \mathbb{E}_{f,x\sim P_S}[E_{\text{out}}(x, R', f)]]]. \tag{2.5}$$

These results correspond to the case where $N_R = 100$.

The matrix is organized placing families of distributions together, with increasing order of standard deviation/time constant. The result that immediately stands out is that there is a significant number

of entries where more than 50% of the runs have better performance when mismatched distributions are used, as indicated by the yellow, orange, and red regions, which constitute 25.8% of all combinations of the probability distributions used.

A number of interesting patterns are worth noting in this plot. The first row, which corresponds to $P_R = U(-1, 1)$, falls under the category of better performance for mismatched distributions for almost any other P_S used. There is also a block structure in the plot, which is no accident due to the way the families of distributions are grouped. Among these blocks, the lower triangular part of the blocks in the diagonal corresponds to cases where the distributions are mismatched but out-of-sample performance is better. We also note that the blocks in the upper-right and lower-left corner show the same pattern in the lower triangular part of the blocks.

Perhaps it is already clear to the reader why this direction of our result is not particularly surprising, and in fact it is not all that significant in practice either. In the setup depicted in this part of the simulation, if we are able to choose a test distribution, then we might as well choose a distribution that concentrates on the region that the system learned best. Such regions are likely to correspond to areas where large concentrations of training data are available. This can be expressed in terms of lower-entropy test distributions, which are over-concentrated around the areas of higher density of training points. Such concentration results in a better *average* out-of-sample performance than that of $P_S = P_R$.

Figure 2.2 illustrates the entropy of different distributions. We plot $H(X_R)$ versus $H(X_S)$, where $H(\cdot)$ is the entropy of the discretized probability distributions and $X_R \sim P_S$ and $X_S \sim P_S$, marking the cases where using $P_S \neq P_R$ resulted in better out-of-sample performance of the algorithm. As it is clear from the plot, these cases occur when $H(X_S) < H(X_R)$.

A simple way to think of the problem is to see that if we could freely choose a test distribution, and our learning algorithm outputs θ^* as the learned parameters that minimizes some loss function $l(x, y, \theta)$ on a training data set $R = \{(x_i, y_i)\}$, then to minimize the out-of-sample error we would choose $P_S(x) = \delta(x - x^\star)$, where δ is the delta-dirac function and $x^\star = \arg\min_R(l(x, y, \theta^*)))$, the point in the input space where the minimum out-of-sample error occurs.

Similar results as those shown in Figure 2.1 are found when $N_R = 300$.

2.1.2 Fixing the test distribution

Figure 2.3 shows the result of the simulation in the other direction. Each entry in the matrix again corresponds to a pair of distributions P_R and P_S. However, this time we fix P_S and evaluate the percentage of runs where using $P_R \neq P_S$ yields better out-of-sample performance than if $P_R = P_S$. More precisely, once again each entry computes the quantity in Equation 2.5.

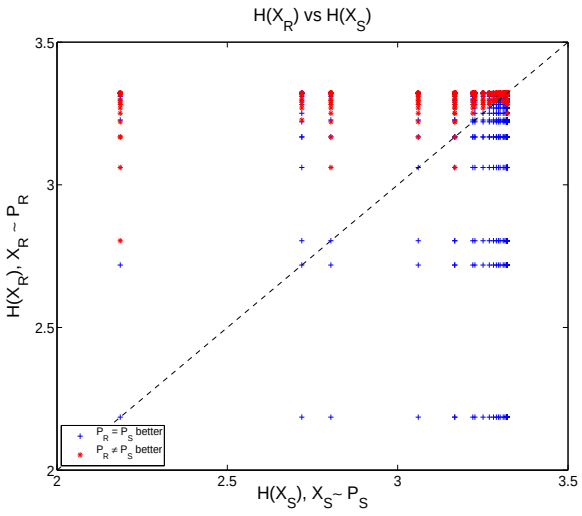

Figure 2.2: $H(X_R)$ vs $H(X_S)$: Characterization of why out-of-sample performance is better if there is a mismatch in distributions when P_R is fixed, using entropy.

This is the case that occurs in practice, where the distribution the system will be tested on is fixed by the problem statement. However, the training set might have been generated with a different distribution, and we would like to determine if training with a data set coming from P_S would have resulted in better out-of-sample performance. If the answer is yes, then one can consider the matching algorithms that we mentioned to transform the training set into what would have been generated using the alternate distribution.

The simulation result is quite surprising, as once again *there is a significant number of entries where more than 50% of the runs have better performance when mismatched distributions are used.* For 14% of the entries, a mismatch between P_R and P_S results in lower out-of-sample error, as indicated by the light green, yellow, orange, and red entries in the matrix.

In this case, although the block structure is still present, there is no longer a clear pattern relating the entropies of the training and test distributions that allows explaining the result easily as in the previous simulation. Notice that there are cases where the mismatch is better if we choose P_R of both lower and higher entropy than the given P_S. This is clear in the plot since the indicated regions in the block structure are no longer lower-triangular but occupy both sides of the diagonal. We look at this result further in the following section, when we analyze the other learning setting: regression.

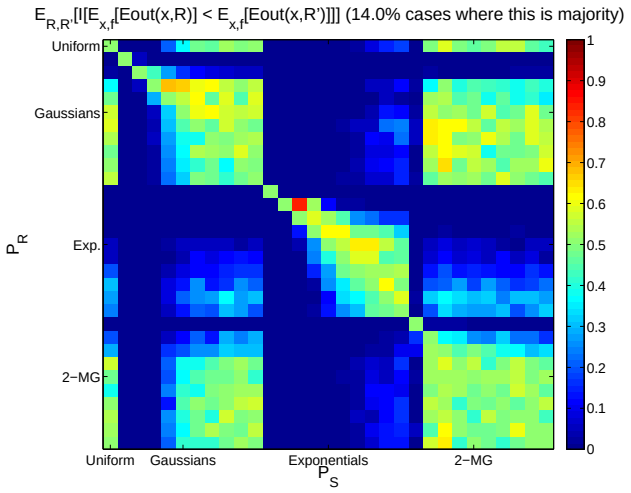

Figure 2.3: Summary of Monte Carlo Simulation. Plot indicates, for each combination of probability distributions, $\mathbb{E}_{R\sim P_R, R'\sim P_S}[I[\mathbb{E}_{f,x\sim P_S}[E_{\text{out}}(x,R,f)] < \mathbb{E}_{f,x\sim P_S}[E_{\text{out}}(x,R',f)]]]$.

2.2 Empirical and analytic results in the regression setting

We have shown empirical evidence that a mismatch in distributions can lead to better out-of-sample performance in the classification setting, and now we focus on the regression setting to cover the other major class of learning problems. In this section, we use the expressions for the expected out-of-sample error as a function of x, a general test point in the input space $\mathcal{X}$, and R, the training set, averaging over target functions and noise realizations. These expressions are derived in detail in Appendix A, for the case where we use a squared loss function and a linear model with non-linear transformations for the hypothesis set. This correspond to the choice of linear model and loss function of the simulations shown in the previous section.

The difference now is that although we choose again $\mathcal{X} = [-1, 1]$, in the regression setting $\mathcal{Y} = \mathbb{R}$. To analyze the most general regression case, we also introduce both "stochastic" and "deterministic" noise [2]. We take $y_i = f(x_i) + \epsilon_i$, where ϵ_i represents the stochastic noise, and where f is more complex than the elements of $\mathcal{H}$, so $f \notin \mathcal{H}$, hence the deterministic noise. We make the usual assumption about the stochastic noise, which is that it has zero mean and is iid. That is, $\mathbb{E}[\epsilon] = 0$, and $\mathbb{E}[\epsilon\epsilon^T] = \sigma_N^2 I$, where I is the identity matrix and σ_N is the standard deviation of the noise. We also make the assumption that the coefficients of the target function that are not included in the model, θ_C, have covariance matrix $\mathbb{E}[\theta_C \theta_C^T] = \sigma_C^2 I$.

As introduced in Appendix A, for simplicity we let

$$z = \phi(x). \tag{2.6}$$

We reorganize the features in z and elements of θ as

$$z^T = [z_M^T \; z_C^T], \quad \theta^T = [\theta_M^T \; \theta_C^T] \tag{2.7}$$

so that the first M features of z correspond to the features in the linear transformation that $\mathcal{H}$ can express. The matrix Z is the "transformed data matrix", with

$$Z = [Z_M \quad Z_C]^T. \tag{2.8}$$

These matrices are precisely defined in Appendix A.

Taking the expected value with respect to the noise, the out-of-sample error at a point $x \in \mathcal{X}$ is given by

$$\mathbb{E}_{f,\epsilon}[E_{\text{out}}(x, R, f, \epsilon)] = \sigma_C^2 \|z_C^T - z_M^T Z_M^\dagger Z_C\|^2 + \sigma_N^2 z_M^T (Z_M^T Z_M)^{-1} z_M + \sigma_N^2 \tag{2.9}$$

Notice that the above expression is independent of θ (i.e., the target function), as well as of the noise. The only remaining randomness in the expression comes from generating R, and from z, the point chosen to test the error, making the analysis very general.

Now, we are interested in minimizing the expected out-of-sample error. Let R denote a training data set generated according to P_R, while R' a data set generated according to P_S. Can we find $P_R \neq P_S$ such that

$$\mathbb{E}_{R,x,\theta_C,\epsilon}[E_{\text{out}}(x, R, f, \epsilon)] < \mathbb{E}_{R',x,\theta_C}[E_{\text{out}}(x, R', f, \epsilon)]? \tag{2.10}$$

The simulation shown in Section 2.1.2, although in a classification setting, suggests that this is the case. We run the same Monte Carlo simulation in this regression setting. The advantage is that the closed-form expression in Equation 2.9 already averages over target functions and noise, allowing us to run in a shorter time more combinations of P_R and P_S. This expression only requires running Monte Carlo simulations for the matrix Z and hence the two terms involving it, $Z_M^\dagger Z_C$ an $(Z_M^T Z_M)^{-1}$. The expectation over $x \sim P_S$ can be done using numerical integration, which is faster than the Monte Carlo simulation in this one-dimensional setting. In this case, we consider the same families of distributions, but we vary the standard deviation of the distribution in smaller steps to obtain a finer grid.

Figure 2.4 indicates that the question posed in Equation 2.10 has an affirmative answer in 21% of the $P_R \neq P_S$ combinations that we considered. This particular simulation used the Fourier harmonics

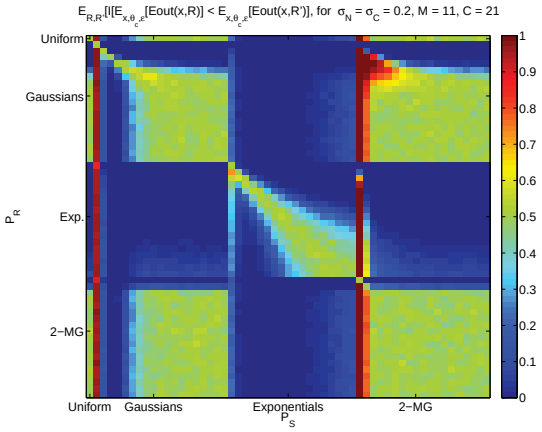

Figure 2.4: Monte Carlo simulation for $\mathbb{E}_{R\sim P_R, R'\sim P_S}\left[I[(\mathbb{E}_{x,\theta_C,\epsilon}[E_{\text{out}}(x,R,\theta,\epsilon)] < \mathbb{E}_{x,\theta_C,\epsilon}[E_{\text{out}}(x,R',\theta,\epsilon)]]\right]$, $M = 11$, $C = 21$, $N = 500$, and $\sigma_N = \sigma_C = 0.2$.

for the non-linear transformation up to order 5, so that $M = 11$. That is,

$$\phi_M(x) = [1 \quad \cos(\pi x) \quad \sin(\pi x) \quad \cdots \quad \cos(5\pi x) \quad \sin(5\pi x)]^T. \tag{2.11}$$

On the other hand, the target functions were generated using harmonics up to order 10, so that $C = 21$, with random Fourier coefficients. Both $\sigma_C = \sigma_N = 0.2$, and $N = 500$. Each entry in the matrix computes

$$\mathbb{E}_{R\sim P_R, R'\sim P_S}\left[I[(\mathbb{E}_{x,\theta,\epsilon}[E_{\text{out}}(x,R,\theta,\epsilon)] < \mathbb{E}_{x,\theta,\epsilon}[E_{\text{out}}(x,R',\theta,\epsilon)]]\right], \tag{2.12}$$

which is the same quantity as that of Equation 2.5, except that now f is determined by θ.

Notice that, as shown in Figure 2.3, the cases where mismatched distributions outperform matched ones cannot be explained using an entropy argument, as was the case in Section 2.1.1. Notice also that there are now combinations for P_R and P_S where almost 100% of the simulations returned lower out-of-sample error for mismatched distributions. In particular, this happened when P_S was a truncated Gaussian with small standard deviation ($\sigma = 0.2$), and when P_S was a mixture of two Gaussians with $\sigma = 0.2$. In addition, we note the similarity between this simulation and the one shown for the classification setting in Figure 2.3.

We varied the size of N in order to see the effect of the sample size. We see very little variation in the results. Holding the other parameters constant, we obtain a very similar result. For $N = 1000$ and for $N = 3000$, we obtain an affirmative answer to the question posed in Equation 2.10 in 21% and 20% of the cases where $P_R \neq P_S$ respectively, so the result does not change from what we obtained in

the $N = 500$ case. For $N = 100$, the percentage is even higher, at 30%. Hence, there is clear evidence that although the number of combinations of distributions for which a mismatch between training and test distributions is larger for smaller N, the result still holds as N grows. Notice that in the simulations, the target function has 21 parameters. Hence, roughly for $N = 100$ there are effectively 5 samples per parameter, while for $N = 3000$ there are 150 samples per parameter. This covers a wide range, from small to large sample sizes, given the complexity of the target function.

Going back to the derived expressions, a closed-form solution for the expected out-of-sample error is given by

$$\mathbb{E}[E_{\text{out}}(x, R, \theta, \epsilon)] = \mathbb{E}_R \int_{-\infty}^{\infty} \sigma_C^2 \|z_C^T - z_M^T Z_M^\dagger Z_C\|^2 P_S(x)dx + \int_{-\infty}^{\infty} \sigma_N^2 z_M^T (Z_M^T Z_M)^{-1} z_M P_S(x)dx + \sigma_N^2. \tag{2.13}$$

It cannot be further reduced analytically due to the inverse matrix terms. Yet, if we assume $C = M$ so that only stochastic noise is present, the expression reduces to

$$\begin{aligned} \mathbb{E}_{\epsilon,R,x,\theta}[E_{\text{out}}(x, R, \theta, \epsilon)] &= \sigma_N^2 + \mathbb{E}_R \int_{-\infty}^{\infty} \sigma_N^2 z^T (Z^T Z)^{-1} z P_S(x)dx \\ &\geq \sigma_N^2 \left(1 + \int_{-\infty}^{\infty} z^T (\mathbb{E}_R[Z^T Z])^{-1} z P_S(x)dx\right), \end{aligned} \tag{2.14}$$

where we use the result in [37] for the expected value of the inverse of a matrix. With this expression, we can find a specific example of a mismatched training distribution that leads to better out-of-sample results. Again, without loss of generality, we pick the linear transformation consisting of Fourier harmonics, namely

$$z = [1 \quad \cos(\pi x) \quad \sin(\pi x) \quad \cdots \quad \cos(m\pi x) \quad \sin(m\pi x)]^T \tag{2.15}$$

as this allows a vast representation of target functions. Here, $M = 2m + 1$. A few examples of the variety of the target functions that can be achieved with this model are shown in Figure 2.5.

If P_R is a Uniform distribution over $\mathcal{X}$, or a Gaussian distribution truncated to this interval, then

$$\begin{aligned} \mathbb{E}_R[Z^T Z] &= \mathbb{E}_R \sum_{i=1}^{N} z_i z_i^T \\ &= N\text{diag}(1, 0.5, 0.5, \ldots, 0.5) \end{aligned} \tag{2.16}$$

The above result is trivial for the uniform distribution case, and can be easily evaluated with numerical

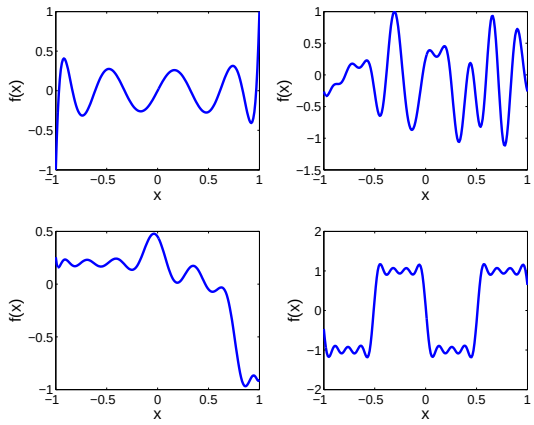

Figure 2.5: Sample realizations of targets generated with a truncated Fourier Series of 10 harmonics.

integration for the truncated Gaussians. This implies that

$$\begin{aligned}\mathbb{E}_{\epsilon,R,x,\theta}[E_{\text{out}}(x,R,\theta,\epsilon)] &\geq \sigma_N^2\left(1+\mathbb{E}_x\left[\frac{2m+1}{N}\right]\right)\\ &= \sigma_N^2\left(1+\frac{M}{N}\right)\end{aligned} \tag{2.17}$$

Now instead, pick R to be distributed according to Uniform$[-a,a]$. In this case,

$$\mathbb{E}_R[Z^TZ]_{ij} = \begin{cases} \text{sinc}(ja) & \text{if } i=1, j \text{ is even}\\ \text{sinc}(ia) & \text{if } j=1, i \text{ is even}\\ 1/2\left(1+(-1)^i\,\text{sinc}(ia)\right) & \text{if } i=j\neq 1\\ 1/2\left(\text{sinc}((i+j)a)+\right. & \text{if } i\neq j, \text{and}\\ \quad\left.\text{sinc}((i-j)a)\right. & i \text{ and } j \text{ odd}\\ 1/2(\text{sinc}\left((i+j)a\right)- & \text{if } i\neq j, \text{and}\\ \quad\text{sinc}((i-j)a) & i \text{ and } j \text{ even}\\ 0 & \text{else}\end{cases} \tag{2.18}$$

Figure 2.6 shows the closed-form bound for various choices of a and $M=10$, choosing P_S to be a truncated Gaussian with $\sigma=0.4$. The dotted line shows the bound for the case $P_R=P_S$. As it is clear from the plot, there are various choices for a so that equation 2.10 is satisfied in terms of the bound.

Since this is only a lower bound on the error, we verify that the minimum suggested by the bound does correspond to a superior mismatched distribution. We Monte-Carlo the value for both

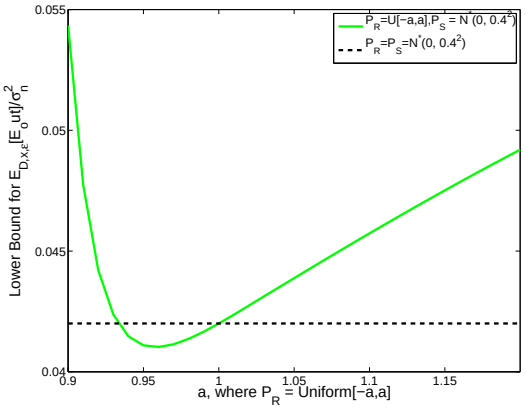

Figure 2.6: Bound for $\mathbb{E}_{R,x,\epsilon}[E_{\text{out}}(x,R)] - \sigma_N^2$ when R is generated with $P_R = P_S = \mathcal{N}^*(0, 0.4^2)$ and for $P_R \neq P_S$ with $P_R = \text{Uniform}[-a, a]$.

cases considered: we choose $P_S = \mathcal{N}^*(0, 0.4^2)$ and generate R' according to P_S, while R is generated according to $U[-0.97, 0.97]$. Notice that we use $a = 0.97$ as this choice results in the lowest error bound from Figure 2.6. Using $m = 10$, $N = 500$ and averaging over 10^8 realizations of R and R' we obtain

$$\mathbb{E}_{R,x,\theta,\epsilon}[E_{\text{out}}(x,R,\theta,\epsilon)] = 1.0429\sigma_N^2 \quad < \quad \mathbb{E}_{R',x,\theta,\epsilon}[E_{\text{out}}(x,R',\theta,\epsilon)] = 1.0440\sigma_N^2 \tag{2.19}$$

Hence, we have a concrete example of a distribution P_R that is different from P_S (Figure 2.7) that leads to better out-of-sample performance, averaging over noise realizations and target functions. The existence of such distributions leads to the concept of a dual distribution which we examine in the next chapter.

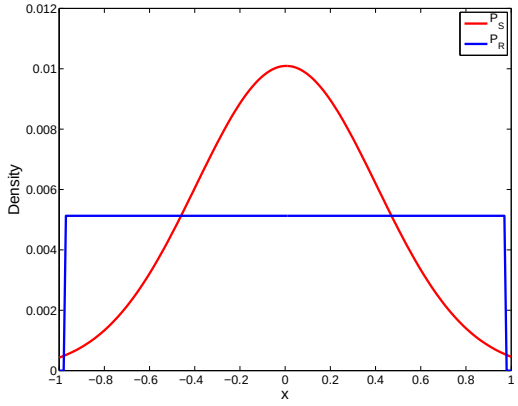

Figure 2.7: Pair of distributions $P_R \neq P_S$ such that expected out-of-sample error is lower when R is generated according to P_R rather than according to P_S for a regression problem in the domain $\mathcal{X} = [-1, 1]$.