Chapter 3

The dual distribution

As shown in the previous chapter, contrary to the common presumption, the optimal distribution from which to sample training data is not necessarily the test distribution P_S. Instead, we call the optimal training distribution the *dual distribution*. This distribution only depends on the test distribution and not in the particular target function in question. In this chapter, we define the dual distribution precisely and then show how to obtain it in the general case, as well as in a practical scenario. We end the chapter with a comparison of the dual distribution approach and a related concept in active learning.

Given a distribution P_S, we define a *dual distribution* $P_R^\star$ to be a distribution that achieves

$$\min_{P_R} \mathbb{E}_{R,x,f,\epsilon}[E_{\text{out}}(x, R, f, \epsilon)] \tag{3.1}$$

where R is a data set generated according to P_R and $x \sim P_S$. The above minimization problem of course has the constraint that P_R must be non-negative and should be normalized, so that the solution yields a valid probability distribution.

3.1 Discrete input spaces

We first find the dual distribution in the case where the input space $\mathcal{X}$ is a discrete set. Let $\mathcal{X} = \{x_j\}_{j=1}^d$, so that P_R and P_S become probability mass functions on d points. Hence, in this setting, finding the dual distribution becomes an optimization problem in $d-1$ dimensions. We only optimize with respect to $d-1$ elements of P_R, since the last element can be determined from the normalization constraint.

For simplicity, we illustrate the solution for a regression problem where only stochastic noise is present. Given R, from Equation 2.9 we can compute the expected out-of-sample error with respect to P_S, the noise, and the target function as

$$\mathbb{E}_{x,\epsilon,\theta}[E_{\text{out}}(x,R,\epsilon,\theta)] = \sigma_N^2 \sum_{i=1}^{d} z_i^T (Z^T Z)^{-1} z_i P_S(x_i). \tag{3.2}$$

In this case, there are $\sum_{i=1}^{N} \binom{d}{i}$ possible data sets of size N (allowing for repetition of points in the data set) that could be obtained for any given P_R. To simplify the notation, since $\mathcal{X}$ is finite, we assign each of the points a number, from 1 to d, and we denote the out-of-sample error for each of these data sets as $E_{i_1,i_2,\cdots,i_N}$, where i_k indicates the element number in $\mathcal{X}$ that corresponds to the k'th data point in R.

Hence, we can find the expected out-of-sample error with respect to P_R as

$$\mathbb{E}_{R,x,\epsilon,\theta}[E_{\text{out}}(x,R,\epsilon,\theta)] = \sum_{i_1,i_2,\ldots,i_N} p_{i_1} p_{i_2} \cdots p_{i_N} E_{i_1,i_2,\ldots,i_N}, \tag{3.3}$$

where all the $E_{i_1,\ldots,i_N}$ can be found using Equation 3.2. Therefore, $P_R^\star$ is the solution to the following optimization problem:

$$\begin{aligned} \min_{p_1,p_2,\ldots,p_d} \quad & \sum_{i_1,i_2,\ldots,i_N} p_{i_1} p_{i_2} \cdots p_{i_N} E_{i_1,i_2,\ldots,i_N} \\ \text{subject to} \quad & \sum_{i=1}^{d} p_i = 1 \\ & p_i \geq 0 \end{aligned} \tag{3.4}$$

Let us look at a concrete example, with $N = 3$,

$$\begin{aligned} z = \Phi(x) &= [\cos(\pi x)\ \ \sin(\pi x)]^T \\ \mathcal{X} &= \{-3/4, -1/4, 0, 1/4, 3/4\} \\ P_S &= [1/3, 0, 1/3, 1/3, 0] \\ [x_1, x_2, x_3, x_4, x_5] &= [-3/4, -1/4, 0, 1/4, 3/4] \end{aligned} \tag{3.5}$$

Solving the optimization problem given in Equation 3.4 yields $P_R^\star \neq P_S$, with

$$P_R^\star = [0.4672, 0.1140, 0.1140, 0.000, 0.3048]. \tag{3.6}$$

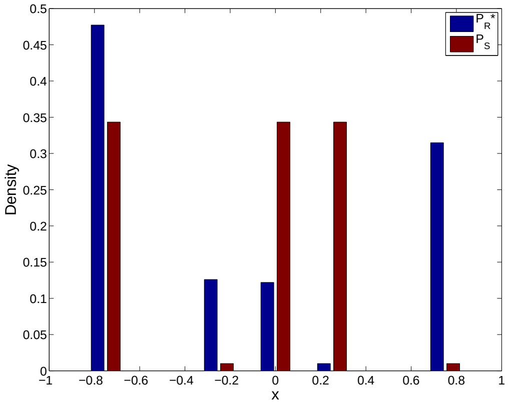


Figure 3.1: Probability mass functions for a given P_S and its dual $P_R^\star$, in a regression problem with stochastic noise, discrete input space $\mathcal{X} = \{-3/4, -1/4, 0, 1/4, 3/4\}$, and $N = 3$.

For this example,

$$\mathbb{E}_{R,x,\epsilon,\theta}[E_{\text{out}}(x, R, \theta, \epsilon)] = 1.1391\sigma_n^2 < \mathbb{E}_{R',x,\epsilon,\theta}[E_{\text{out}}(x, R, \theta, \epsilon)] = 1.5778\sigma_n^2, \tag{3.7}$$

where R' is generated according to P_S and R according to $P_R^\star$. Clearly there is a gain by training with the dual distribution, in this case. When running the optimization, for data sets that have repeated points that result in undefined out-of-sample error as the matrix $(Z^TZ)^{-1}$ is singular, we conservatively take their error to be the maximum finite out-of-sample error over all combinations of possible data sets. Figure 3.1 shows the dual distribution found, along with the given P_S.

Notice that if a different loss function is chosen and no closed form solution exists for $E_{\text{out}}(x, R)$, the dual distribution can still be found using the same procedure as above. The only difference is that $E_{\text{out}}(x, R)$ must be estimated, using a held-out set for instance, for each possible dataset R, so that the corresponding $E_{i_1,\ldots,i_N}$ can be computed and given as inputs to the optimization problem of Equation 3.4

A very important property of the optimization problem formulated in Equation 3.4 is that it is a convex optimization program. In fact it is a Geometric Program, although different from a standard Geometric Program, since the equality constraint is not a monomial. Yet, the problem is still convex. To illustrate this, let

$$\begin{aligned} \psi_i &= \log(p_i) & (3.8)\\ \Lambda_{i_1,\ldots,i_N} &= \log(E_{i_1,\ldots,i_N}). & (3.9) \end{aligned}$$

This change of variables implicitly makes $p_i > 0$ so that the inequality constraints can be removed. Also, the problem can be rewritten as

$$\begin{aligned} \min_{\psi_1,\psi_2,\ldots,\psi_d} \quad & \sum_{i_1,i_2,\ldots,i_N} e^{\sum_{k=1}^{N} \psi_{i_k} + \Lambda_{i_1,i_2,\ldots,i_N}} && (3.10) \\ \text{subject to} \quad & \sum_{i=1}^{d} e_i^{\psi} = 1 \\ & && (3.11) \end{aligned}$$

Notice that the objective function is a sum of exponential functions of affine functions of ψ_i. Since exponential functions are convex, affine transformations of convex functions are also convex, and sums of convex functions result in a convex function, the objective in Equation 3.10 is convex [19]. Following the same argument, the equality constraint is also convex, so that the optimization problem is a convex program.

Hence, if a minimum is found, this is the global optimum with a corresponding dual distribution. This problem can be solved with any convex optimization package. Furthermore, in most applications, P_S is generally unknown and is estimated by binning the data, which leads to a discrete version of P_S. Therefore, this discrete formulation is appropriate to find dual distributions in such settings. Solving the Geometric Program described by Equation 3.10 thus allows us to find the dual distribution in various practical settings.

Nevertheless, we need to address the more general case of continuous input spaces. The following section describes how to find the dual distribution in that case, as well as how to implement it in a practical scenario.

3.2 The continuous case

When the input space $\mathcal{X}$ is continuous, as it is the case in most applications, the optimization problem in Equation 3.1 is a functional optimization problem, since we are interested in finding the full distribution P_R. We denote the corresponding probability density function by p_R, and optimize with respect to this density. The objective function of the optimization problem can be written as the functional $J : \mathcal{P} \to \mathbb{R}$

$$J(p) = \int_{x_N} \cdots \int_{x_1} L(x_1, \ldots, x_N) \prod_{i=1}^{N} p(x_i) dx_1 \cdots dx_N, \tag{3.12}$$

where

$$L(x_1, \ldots, x_N) = \mathbb{E}_{x \sim P_S, f, \epsilon}[E_{\text{out}}(x, R, f, \epsilon)], \tag{3.13}$$

and $\mathcal{P}$ is the set of all probability density functions with $\mathcal{P} \subset L^1$. (Recall an L^p space over X is defined as the space of functions f for which $\int_X |f(x)|^p < \infty$. Since probability density functions integrate to unity, they are elements of L^1). In the following subsection, we use functional calculus to arrive at the analytic condition that the dual distribution must satisfy.

3.2.1 Analytic condition for the dual distribution

To minimize the functional $J(p)$, we first transform the variables, as we did in Section 3.1. Let

$$\psi(x) = \log p(x) \tag{3.14}$$

$$\Lambda(x_1, \ldots, x_N) = \log(L(x_1, \ldots, x_N)). \tag{3.15}$$

The optimization problem becomes

$$\min_{\psi} = J(\psi) \tag{3.16}$$

$$\text{subject to} \quad \int e^{\psi(x)} dx = 1$$

where

$$J(\psi) = \int_{x_N} \cdots \int_{x_1} e^{\Lambda(x_1,\ldots,x_N)) + \sum_{i=1}^{N} \psi(x_i)} dx_1 \cdots dx_N, \tag{3.17}$$

and where the positivity constraints are implicit, given the domain of the logarithm.

Now, recall that the gradient of a functional $J(\psi)$, denoted as $\nabla_\psi J$, is given by [28]

$$J(\psi + \delta\zeta) = J(\psi) + \delta\langle \nabla_\psi J, \zeta \rangle + \mathcal{O}(\delta^2), \tag{3.18}$$

where $\delta \in \mathbb{R}, \delta > 0$, and $\zeta \in \mathcal{P}$ is an arbitrary function. Consider the Lagrangian

$$\mathcal{L}(\psi) = J(\psi) + \lambda \left(\int e^{\psi(x)} dx - 1 \right). \tag{3.19}$$

Then, the dual distribution must satisfy

$$\nabla_\psi(\mathcal{L}(\psi(x))) = 0. \tag{3.20}$$

In fact, we can use the Euler-Lagrange theorem [30] to show that if there is a function ψ that satisfies Equation 3.20, then it is the global minimizer. The theorem states that for a function $f \in \mathcal{C}^2$, with $f : [a, b]^d \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ where $\mathcal{C}^2$ denotes continuously twice differentiable functions, and we have the functional

$$\inf_{\mathbb{X}} I(u) = \inf_{\mathbb{X}} \int_{\mathcal{X}} f(x, u, u')dx \tag{3.21}$$

where $\mathbb{X} = \{u \in \mathcal{C}^1, u : [a, b]^d \rightarrow \mathbb{R}, |u|_{\partial_1 \Omega} = u_0\}$, u_0 are the boundary conditions, and $\mathcal{X} = [a, b]^d$, then if $I(u)$ admits a minimizer $\bar{u} \in \mathcal{C}^2$, then $\bar{u}$ satisfies the Euler-Lagrange (EL) equation:

$$\sum \frac{\partial}{\partial x_i} \frac{\partial}{\partial u'} f(x, \bar{u}(x), \bar{u}'(x)) - \frac{\partial}{\partial u} f(x, \bar{u}(x), \bar{u}'(x)) = 0. \tag{3.22}$$

Conversely, if $\bar{u}$ satisfies the EL equation and the mapping $M(u, u') \rightarrow f(x, u, u')$ is convex for every $x \in [a, b]^d$, then $\bar{u}$ minimizes $I(u)$.

In our case, $u = \psi$, and $J(\psi) = I(\psi)$. Also, in our case, u' does not appear, so that $f(x, \psi, \psi') = f(x, \psi)$. Hence, having the gradient of the Lagrangian with respect to ψ equal to 0 is equivalent to satisfying the EL equation. This is the necessary condition.

Now, we can show that it is in fact a sufficient condition by using the converse. Notice that in our case $\mathbb{X} = \mathcal{P}$ is a convex set, as convex combinations of density functions are also convex. Hence, all that remains to show is that the mapping M is convex, that is, show that for $0 \leq \alpha \leq 1$, $\alpha \in \mathbb{R}$,

$$M(\alpha\psi + (1 - \alpha)\phi) < \alpha M(\psi) + (1 - \alpha)M(\phi). \tag{3.23}$$

Substituting, we have in the left hand side,

$$e^{\Lambda(x_1,\ldots,x_N)+\alpha \sum_i \psi(x_i)+(1-\alpha) \sum_i \phi(x_i)}. \tag{3.24}$$

On the right hand side we have

$$\alpha e^{\Lambda(x_1,\ldots,x_N)+\sum_i \psi(x_i)} + (1 - \alpha)e^{\Lambda(x_1,\ldots,x_N)+\sum_i \phi(x_i)}. \tag{3.25}$$

Now, we notice that due to the strict convexity of the exponential function

$$e^{\alpha\theta_1+(1-\alpha)\theta_2} < \alpha e^{\theta_1} + (1 - \alpha)e^{\theta_2}. \tag{3.26}$$

Hence, dividing both sides of Equation 3.23 by $e^{\Lambda(x_1,\ldots,x_N)}$ and substituting $\theta_1 = \sum_i \psi(x_i)$ and $\theta_2 = \sum_i \phi(x_i)$ shows that the mapping M is strictly convex.

This implies that if the dual distribution exists, that is, if we find ψ that satisfies Equation 3.20, then it is the unique, and is the global minimizer of J, and by constraint satisfaction, also the minimizer of $\mathcal{L}$. Note the theorem assumes continuous differentiability of u, but the theorem can be generalized for functions that are continuously differentiable, except at sets of measure zero.

We now compute the gradient of the Lagrangian. For simplicity, let dR denote $dx_1 \cdots dx_N$, and let $\mathcal{R}$ denote the support of the set $\{x_1, \ldots, x_N\}$ then

$$\begin{aligned} J(\psi+\delta\xi) &= \int_{\mathcal{R}} e^{\sum_{i=1}^N \psi(x_i)+\delta\xi(x_i)+\Lambda(x_1,\ldots,x_N)} dR \\ &= \int_{\mathcal{R}} e^{\sum_{i=1}^N \psi(x_i)+\Lambda(x_1,\ldots,x_N)} \left(1+\delta\sum_{i=1}^N \xi(x_i)+\mathcal{O}(\delta^2)\right) dR \\ &= J(\psi)+\delta\int_{\mathcal{R}} e^{\sum_{i=1}^N \psi(x_i)+\Lambda(x_1,\ldots,x_N)}\sum_{i=1}^N\xi(x_i)dR \\ &= J(\psi)+\sum_{i=1}^N \left\langle \int_{x_n,n\neq i} e^{\sum_{i=1}^N \psi(x_i)+\Lambda(x_1,\ldots,x_N)} dx_{n_{n\neq i}}, \xi(x_i)\right\rangle, \end{aligned} \tag{3.27}$$

where the simplification follows from using a Taylor expansion of the exponential. Finally, since the loss functions we are interested in are independent of the order of the points in the training set, then the logarithm of the loss function $\Lambda(x_1, \ldots, x_N)$ is symmetric with respect to x_i. Therefore,

$$\nabla_\psi(J(\psi(x_n))) = N\mathbb{E}_{\substack{x_i\sim e^\psi\\ i\neq n}}[L(x_1,\ldots,x_N)]. \tag{3.28}$$

Following a similar procedure for the second term in the Lagrangian, we obtain that at point x_n

$$\nabla_\psi(\mathcal{L}(\psi(x_n)) = \left(N\mathbb{E}_{\substack{x_i\sim e^\psi\\ i\neq n}}[L(x_1,\ldots,x_N)]+\lambda\right)e^{\psi(x_n)} \tag{3.29}$$

We can now use the constraint to find λ by integrating the above equation over x_n. We obtain

$$\lambda = -Ne^{\psi(x_n)}\mathbb{E}_{\substack{x_i\sim p\\ i=1,\ldots,N}}[L(x_1,\ldots,x_N)] \tag{3.30}$$

Substituting for λ we obtain the optimality condition that the dual distribution needs to satisfy:

$$\boxed{p(x_n)\left(\mathbb{E}_{\substack{x_i\sim p\\ i\neq n}}[L(x_1,\ldots,x_N)]-\mathbb{E}_{\substack{x_i\sim p\\ i=1,\ldots,N}}[L(x_1,\ldots,x_N)]\right)=0.} \tag{3.31}$$

This condition applies to the dual distribution in the general case, without making assumptions about the target class or the learning model. Now, all that remains is to find p that satisfies this condition, which can be done, for example, using functional gradient descent [49].

The functional gradient descent step is given by

$$p(x) := p(x) - \eta\nabla(\mathcal{L}(p(x)) \tag{3.32}$$

where η is the learning rate, hence

$$p(x_n) := p(x_n) - \eta N \left(\mathbb{E}_{\substack{x_i \sim p \\ i \neq n}}[L(x_1, \dots, x_N)] - \mathbb{E}_{\substack{x_i \sim p \\ i=1,\dots,N}}[L(x_1, \dots, x_N)] \right) p(x_n). \tag{3.33}$$

Notice that the integral of the update over x_n is 0. Hence, this update guarantees that the normalization constraint is satisfied at each step, so that gradient descent works in this case as in an unconstrained problem. Therefore, if the initial condition is a valid probability density function (pdf), all subsequent p's will also be valid pdf's.

The interpretation of this update is very intuitive: If a point x_n is included in the training set, and the resulting out-of-sample error is lower than the expected out-of-sample error with N points, that is $\mathbb{E}_{\substack{x_i \sim p \\ i \neq n}}[L(x_1, \dots, x_N)] < \mathbb{E}_{\substack{x_i \sim p \\ i=1,\dots,N}}[L(x_1, \dots, x_N)]$, then $p(x_n)$ should be increased. If including the point leads to a higher out-of-sample error, then the density at this point should be decreased.

In the following subsection, we introduce a concrete example of how the condition of Equation 3.31 can be used computationally to derive the dual distribution.

3.2.2 Dual distribution examples

As shown in the previous subsection, finding the dual distribution reduces to performing functional gradient descent. However, the update rule depends on being able to compute the expected out-of-sample error $\mathbb{E}_{\substack{x_i \sim p \\ i \neq n}}[L(x_1, \dots, x_N)]$. Computing the expected value with respect to the training set can be readily done using Monte Carlo (MC) simulation. This can be slow unless a closed form for $L(x_1, \dots, x_N)$ exists.

If a squared loss function is used for ℓ, and the hypothesis class $\mathcal{H}$ is chosen to be a linear model (which can include non-linear transformations of the inputs), then a closed-form solution for $L(x_1, \dots, x_N)$ exists. This solution is independent of the specific target function. Hence, in this setting, the dual distribution can readily be found. The closed-form solution, as derived in Appendix A is given by

$$L(x_1, \dots, x_N) = \sigma_C^2 \|\phi_C(x)^T - \phi_M(x)^T \Phi_{MM}^{-1} \Phi_{MC}\|^2 + \mathbb{E}_{x \sim P_S}\left[\sigma_N^2 \phi_M(x)^T \Phi_{MM}^{-1} \phi_M(x)\right] + \sigma_N^2, \tag{3.34}$$

where $\phi : \mathcal{X} \to \mathcal{Z}^{M+C}$ denotes the transformation of the input, with

$$\phi(x) = [\phi_M(x)^T \quad \phi_C(x)^T]^T, \tag{3.35}$$

so that $\phi_M : \mathcal{X} \to \mathcal{Z}^M$ represents the part of the target function that can be captured by the model, and $\phi_C(x) : \mathcal{X} \to \mathcal{Z}^C$ is the part of the target that cannot be captured by the model. The matrices

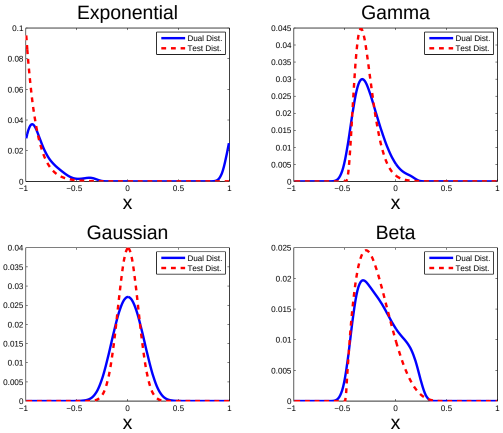


Figure 3.2: Examples of dual distributions, for 1-D test distributions, in a linear regression problem.

Table 3.1: Out-of-sample (OoS) performance improvement when training the learning algorithm with data coming from the dual distribution rather than from the test distribution

Test Distr.	Parameters	OoS error improvement
Exponential	$\lambda = 5$	46.3%
Gamma	$\alpha = 4,\ \beta = 0.05$	32.0%
Gaussian	$\mu = 0,\ \sigma = 0.1$	21.4%
Beta	$\alpha = 2,\ \beta = 5$	10.0%
F	$\nu_1 = 100,\ \nu_2 = 80$	5.7%
Weibull	$\lambda = 1,\ k = 5$	2.2%
Uniform	[-1,1]	0.5%
2-D Gaussian	$\Sigma = [0.1^2\ 0.08; 0.08\ 0.1^2]$	22.6%
2-D MG	$\Sigma = [0.1^2\ 0.06; 0.06\ 0.1^2]$	5.71%

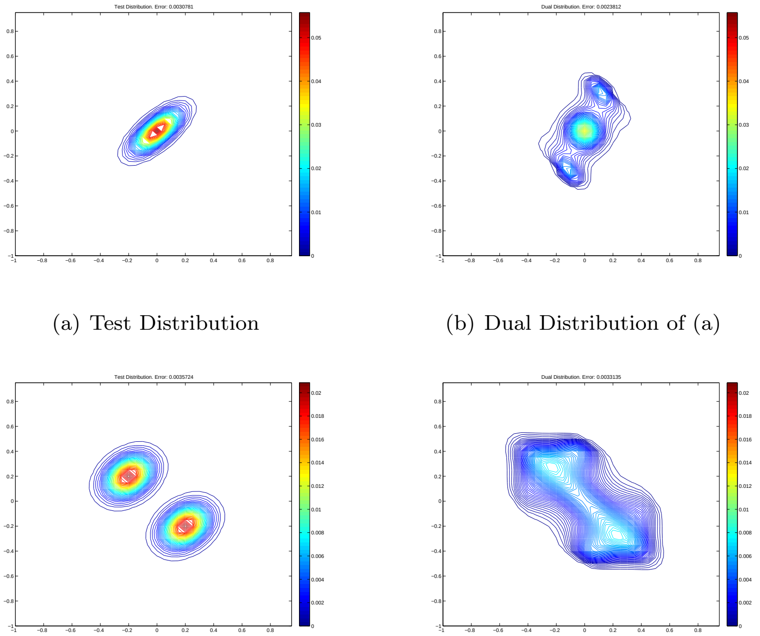
$\Phi_{MM} \in \mathcal{Z}^{M\times M}$ and $\Phi_{MC} \in \mathcal{Z}^{M\times C}$ defined for the training input points $x_1, \ldots, x_N$ are given by

$$\Phi_{MM} = Z_M^T Z_M = \sum_{i=1}^{N} \phi_M(x_i)\phi_M(x_i)^T, \tag{3.36}$$

$$\Phi_{MC} = Z_M^T Z_C = \sum_{i=1}^{N} \phi_M(x_i)\phi_C(x_i)^T. \tag{3.37}$$

Finally σ_N^2 and σ_C^2 characterize the energy of the stochastic noise and ‘excess’ target complexity as explained before.

Figure 3.2 shows the dual distributions for various one-dimensional test distributions for the regression setup. The learning model uses Fourier harmonics of the input, while the target functions are constructed by considering functions that include Fourier harmonics higher than those that belong to



(a) Test Distribution

(b) Dual Distribution of (a)

(c) Test Distribution

(d) Dual Distribution of (c)

Figure 3.3: Examples of dual distribution, for 2-D test distributions. (a) 2-D Gaussian; (b) Dual distribution for (a); (c) Mixture of 2-D Gaussians; (d) Dual distribution for (c).

the model. The simulation parameters were set to $N = 100$, $M = 3$, $C = 5$, $\sigma_N = \sigma_C = 0.2$. The input domain is $\mathcal{X} = [-1, 1]$, so the distributions were zeroed out outside this domain and renormalized. Table 3.1 shows the parameters of the test distributions and also indicates the improvement in out-of-sample performance when the learning algorithm is trained with samples coming from the dual distribution, rather than from the test distribution. Figure 3.3 shows the dual distribution for two-dimensional test distributions.

As it is clear from Table 3.1, the gains in using the dual distribution can be significant. For these examples, N was chosen so that there were enough samples to estimate the three parameters in the model ($M = 3$), and the target was more complex than the model.

The reader may be wondering how the sample size (N), the excess target complexity with respect to the model ($C - M$) and its magnitude (σ_C), and the stochastic noise level (σ_N) affect the dual distribution. We address this question in the following section.

3.3 Variability of the dual distribution

The definition of the dual distribution is based on some specific aspects of the learning problem, such as the training set size, the target complexity, and the model complexity. In this section, we explore the change in dual distribution due to these factors.

3.3.1 Asymptotic behavior

The first factor we analyze is the dependence of the dual distribution on N, the training set sample size. In particular, consider the case where $N \to \infty$. Recall that the dual distribution is the distribution P that minimizes the quantity $\mathbb{E}_{x_i \sim P}[L(x_1, \ldots, x_N)]$. Using the closed-form expression for $L(x_1, \ldots, x_N)$ in the squared loss and linear model with non-linear transformations case, we can separate the impact of the stochastic and deterministic noise terms. The stochastic term, that is, the term proportional to σ_N^2 from Equation 3.34, is $\mathcal{O}(1/N)$. Notice that:

$$\mathbb{E}_{x_i \sim P}\left[\Phi_{MM}^{-1}\right] = \frac{1}{N}\mathbb{E}_{x_i}\left[\left(\frac{1}{N}\sum_{i=1}^{N}\phi_M(x_i)\phi_M(x_i)^T\right)^{-1}\right]$$

As $N \to \infty$,

$$\frac{1}{N}\sum_{i=1}^{N}\phi_M(x_i)\phi_M(x_i)^T \xrightarrow{P} \mathbb{E}_{x_i \sim P}[\phi_M(x_i)\phi_M(x_i)^T] \tag{3.38}$$

where $\xrightarrow{P}$ denotes convergence in probability. Substituting, the stochastic noise term simplifies to

$$\frac{1}{N}\mathbb{E}_x\left[\sigma_N^2\phi_M(x)^T\mathbb{E}_{x_i}\left[\phi_M(x_i)\phi_M(x_i)^T\right]^{-1}\phi_M(x)\right]. \tag{3.39}$$

Therefore, this term vanishes as $N \to \infty$.

The remaining term, on the other hand, is $\mathcal{O}(1)$, so this is the term that must be minimized. Following a similar analysis as above, it follows that

$$\lim_{N\to\infty}\mathbb{E}_{x_i \sim P}\left[L(x_1, \ldots, x_N)\right] = \sigma_N^2 + \sigma_C^2\mathbb{E}_x\left[\|\phi_C(x)^T - \phi_M(x)^T\Phi\|^2\right], \tag{3.40}$$

where

$$\Phi = \left(\mathbb{E}_{x_i}\left[\phi_M(x_i)\phi_M(x_i)^T\right]\right)^{-1}\mathbb{E}_{x_i}\left[\phi_M(x_i)\phi_C(x_i)^T\right]. \tag{3.41}$$

Notice that if the collection of features $\{\phi_i(x)\}_{i=1}^{M+C}$ (the components of $\phi(x)$) form an orthonormal set under P, then by definition

$$\mathbb{E}_{x_i \sim P}[\phi_i(x)\phi_j(x)] = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}. \tag{3.42}$$

Therefore, $\mathbb{E}_{x_i}\left[\phi_M(x_i)\phi_M(x_i)^T\right] = I$, and $\mathbb{E}_{x_i}\left[\phi_M(x_i)\phi_C(x_i)^T\right] = \mathbf{0}$, so that $\Phi = \mathbf{0}$. This orthonormality condition holds, for example, when P is a Gaussian distribution and the features are Hermite

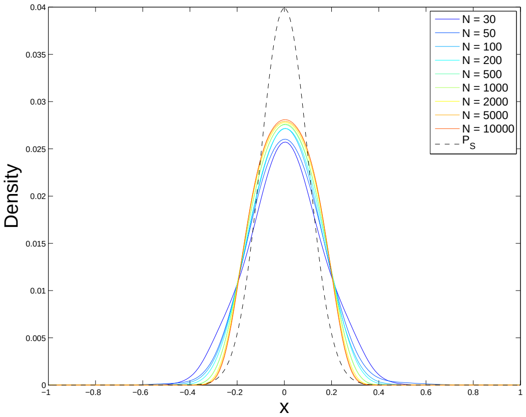


Figure 3.4: Example Dual Distributions in 1-Dimension when the training set size N changes, for $P_S = \mathcal{N}(0, 0.1^2)$, $M = 3$, $C = 5$, using a linear model with Fourier harmonics and a squared loss function.

polynomials, or when P is a uniform distribution and the features are Fourier harmonics, among other cases. In this case, the error would reduce to $\sigma_N^2 + C\sigma_C^2$, and hence there would be no dependence of the error on the training distribution.

However, when the features are not orthonormal under P, the out-of-sample error still changes with the training distribution in the limit as $N \to \infty$. We can minimize Equation 3.40 with respect to Φ. This optimization problem is strictly convex, as it is a quadratic program in the entries of Φ. Finding the gradient and setting it to zero, we find that the necessary and sufficient condition for the minimum is to satisfy the equation

$$\Phi^T \mathbb{E}_{x \sim P_S}[\phi_M(x)] = \mathbb{E}_{x \sim P_S}[\phi_C(x)]. \tag{3.43}$$

The solution to this equation will depend on the type of features chosen. For example, if the features outside the model have mean zero, making the right hand side vanish, then the distribution P that makes the features orthogonal will be the solution.

Figure 3.4 shows the effect of N on the dual distribution in a specific example. For this example $P_S = \mathcal{N}(0, 0.1^2)$, $M = 3$, $C = 5$, $\sigma_N = \sigma_C = 0.2$ and N varies. We used a linear model with Fourier harmonics and a squared loss function. Notice that the variability of the dual distribution is small as N changes.

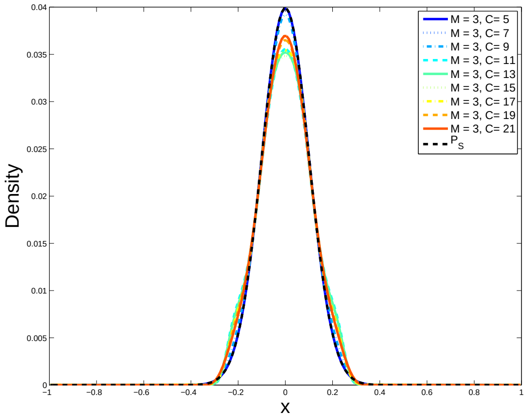


Figure 3.5: Example Dual Distributions in 1-Dimension when the deterministic noise changes, for $P_S = \mathcal{N}(0, 0.1^2)$, $N = 100$, $M = 3$, using a linear model with Fourier harmonics and a squared loss function.

3.3.2 Effect of noise and complexity

We now look at the effect of target complexity. Notice that as the target complexity grows, the deterministic noise term dominates $L(x_1, \ldots, x_N)$. Hence, although the stochastic noise term does not vanish as it is the case when $N \to \infty$, it is still the deterministic noise term that drives the minimization. Figure 3.5 shows the dual distributions for the same test distribution, as the target complexity increases. As the figure shows, there is little variability with respect to the change in target complexity. The variability actually disappears completely if Hermite polynomials are chosen for the features. In this case, for all values of C, $P^\star = P_S$, and Figure 3.6 exemplifies this behavior.

If we now look at the case where only stochastic noise is present in the data, we notice that for finite N, the error becomes

$$\sigma_N^2 \mathbb{E}_{x \sim P_S} \left[\phi_M(x)^T \mathbb{E}_{x_i \sim P} \left[\Phi_{MM}^{-1} \right] \phi_M(x) \right] . \tag{3.44}$$

Again we have a quadratic form, but this time in terms of the matrix Φ_{MM} rather than in term of the matrix Φ. This objective function has a minimum of zero, which is achieved at $\mathbb{E}[\Phi_{MM}] = \mathbf{0}$. However, Φ_{MM} follows the particular form defined in Equation 3.36, which constrains the quadratic program so it yields a different solution.

Figure 3.7 illustrates the effect of increasing the stochastic noise in a concrete example, where the dual distribution is calculated for the same test distribution, and σ_N is increased while holding N, M, and C constant. Notice that for small values of σ_N, $P_R^\star = P_S$.

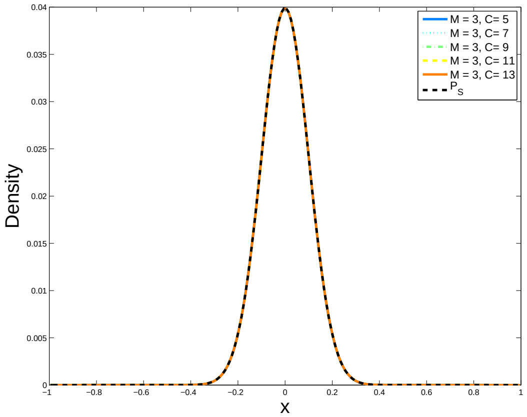


Figure 3.6: Example Dual Distributions in 1-Dimension when the deterministic noise changes, for $P_S = \mathcal{N}(0, 0.1^2)$, $N = 100$, $M = 3$, using a linear model with Hermite polynomial features and a squared loss function.

As it can be seen from the above analysis, the dual distribution is fairly robust with respect to the different components of the learning problem. Namely, the sample size, the noise, and the target complexity. This property allows using the dual distribution in a practical setting where components like the level of noise and target complexity might not be exactly known. The following subsection describes how to find the dual distribution in a practical setting.

3.4 Using the dual distribution in a practical setting

The dual distribution can be applied in two different settings. The first is the population-based active learning setting. This is a special case of active learning, in which contrary to supervised learning where the training data set is fixed, it is possible to sample points according to a desired distribution. This active learning setting is common in applications of experiment design, where the idea is precisely to design the distribution from which points will be sampled. In this case, the design distribution plays the role of the dual distribution, and is chosen by searching within a class of distributions [63].

The second setting where the dual distribution can be used, is in the supervised learning setting case that we have been discussing. In this section, we will show how to use the dual distribution even though the data has already been generated using a fixed distribution. We describe in detail how to do this, and show results on benchmark datasets.

The supervised learning setting poses two challenges for the use of the dual distribution method.

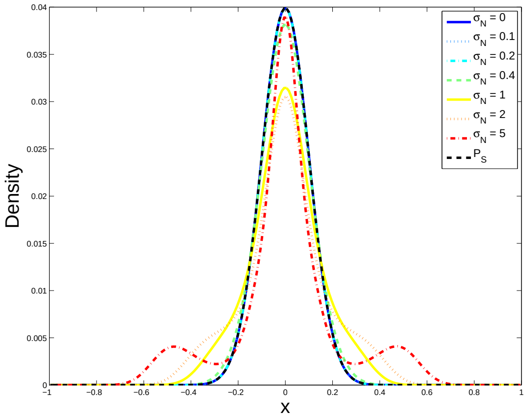


Figure 3.7: Example Dual Distributions in 1-Dimension when the stochastic noise changes, for $P_S = \mathcal{N}(0, 0.1^2)$, $N = 100$, $M = 3$, $C = 5$, using a linear model with Fourier harmonics and a squared loss function.

First, the data set is fixed, so the expected values in Equation 3.33, which are taken with respect to data set generations, cannot be evaluated. Hence, there is a problem in computing the dual distribution itself. The second problem is how to use the dual distribution, since the data is already fixed. It is now necessary to make the data set appear as if it came from a different distribution. Here we describe how to approach both problems.

In order to get the dual distribution with only one data set sample, we make use of the fact that in this setting, the dual distribution only needs to be computed at the positions x_i, $i = 1, \ldots, N$. The reason for this is that we will use matching algorithms to make the sample look as if it was distributed according to the dual distribution. As we explain shortly, matching algorithms only need to compute weights for each of the samples. Hence, it is no longer necessary to compute the full function, but simply its values at N locations. Also, we notice that the gradient at a point x_n is given by the difference in the expected loss with respect to $N-1$ training points, having x_n present in the training set, and the expected loss with respect to N training points (Equation 3.31). So, given that only one data set is available, $\mathbb{E}_{\substack{x_i \sim p \\ i=1,\ldots,N}}[L(x_1, \ldots, x_N)]$ is approximated by the estimate of the loss using a single sample (i.e. a single data set). On the other hand, $\mathbb{E}_{\substack{x_i \sim p \\ i \neq n}}[L(x_1, \ldots, x_N)]$ is estimated by increasing the weight of point x_n and finding the resulting loss. The difference of the two terms will approximate the effect of this point on the loss, and hence determine an approximate value of the gradient at this point.

Once the dual distribution is computed, we need to make the training dataset look as if it was distributed according to this new distribution. To do this, we make use of the available methods from the covariate shift literature. These methods are described in Section 1.2. All these methods are variants of importance weighting [64], and their goal is to estimate the weights $w(x) = p_S(x)/p_R(x)$, where p_R and p_S are the training and test densities, respectively. They do this, in order to match P_R to P_S. Some of the methods, like KMM [38], KLIEP [67], and LSIF [42] usually perform better in the covariate shift correction problem, as they try to estimate the ratio directly, rather than computing the numerator and denominator separately. This can be done when there are unlabeled samples available, coming from both the training and test distributions. In our case, the importance weights are given by $w(x) = p^{\star}_R(x)/p_R(x)$. Here, the numerator is found directly through functional gradient descent, having no available samples distributed according to it. Hence, it is necessary to use methods that actually compute the training density p_R. This can be done either by finding a histogram of the training set with the adequate resolution, or using other non-parametric methods like Kernel Density Estimation (KDE) [56] and [53]. Chapter 5 proposes an alternative method that can be used to match the training distribution to any other distribution, such as the dual. We call this algorithm Soft Matching.

Therefore, the dual distribution can be used in the supervised learning setting, using the mentioned approximation for the functional gradient descent, and making use of importance weighting to change the distribution of the training data. Table 3.2 shows the average out-of-sample error, in both classification and regression tasks on 17 benchmark datasets [7] and [68], when the training set is transformed so that it appears distributed as the dual distribution. The values are compared to the case where no changes are made to the training set.

The results are averaged over 1,000 different splits of the data into training and test sets. The training set is also split further, as a validation set is needed to compute the expected loss for the functional gradient descent. The reported errors are on the test set which is not used at all during training, nor during computation of the dual distribution. For all datasets 25% of the data was left aside for testing, 25% was part of the validation set, and the remaining 50% was used for training. For classification problems, weighted SVMs with Gaussian kernels were used, choosing the kernel width as in [38], with the `libsvm` implementation [23]. Ridge regression was used for the remaining data sets, with regularization parameter $\lambda = 0.1$.

As can be seen from the table, in all of the classification problems, the use of the dual distribution led to a lower out-of-sample error (classification error percentage). Numbers in boldface indicate that the improvement is statistically significant. For the regression problems, the improvements in normalized mean-squared error (NMSE) were smaller but still present. Since the use of weights can lead to an increase in variance or equivalently a sample size reduction [61], it is not surprising that the improvement in performance is lower than in the examples where direct sampling from the dual

Table 3.2: Generalization error in benchmark datasets under the supervised learning paradigm, with and without the use of the dual distribution. 0/1 classification error is reported for the classification tasks; normalized mean-squared error (NMSE) is shown for regression tasks. N is the size of the full data set. All numbers are multiplied by 100.

Dataset	N	Dual	No Dual
(Classif.)		0/1 Error	
Breast C.	278	**25.70 ± 0.14**	27.53 ± 0.15
Breast WI	683	**4.38 ± 0.04**	4.45 ± 0.04
German Cred.	768	**23.96 ± 0.09**	25.08 ± 0.09
Haberman	306	**25.56 ± 0.13**	26.10 ± 0.13
Diabetes	768	**24.09 ± 0.09**	25.08 ± 0.09
Ionosphere	351	**6.28 ± 0.07**	6.41 ± 0.07
(Regression)		NMSE	
Abalone	4177	**50.25 ± 0.10**	50.75 ± 0.10
Ailerons	13750	**18.63 ± 0.02**	18.65 ± 0.02
Bank8FM	8192	**6.70 ± 0.01**	6.72 ± 0.01
Bank32NH	8192	46.84 ± 0.06	46.87 ± 0.06
Bos. Housing	606	36.74 ± 0.27	36.94 ± 0.27
CA Housing	20650	36.15 ± 0.04	36.19 ± 0.04
Cpu-act	8192	**25.97 ± 0.08**	26.39 ± 0.08
Cpu-small	8192	**30.11 ± 0.09**	30.47 ± 0.09
δ-Ailerons	9129	49.81 ± 0.10	49.81 ± 0.10
Kin8nm	8192	58.83 ± 0.05	58.83 ± 0.05
Puma8nh	8192	61.73 ± 0.05	61.73 ± 0.05

distribution is possible as in Table 3.1. However, the key takeaway is that there is empirical evidence that shows that using the dual distribution does improve out-of-sample performance in a supervised learning setting, in both classification and regression problems.

Since all datasets considered have a multidimensional test distribution, the dual distribution was found for each of the projections of the test distribution, along its original coordinates, and the distribution that led to the lowest error in the validation set was chosen in each run. As we discuss in Section 3.5, finding the dual distribution for a multi-dimensional test distribution is computationally more difficult, as it is necessary to compute numerically a function at every point in a high-dimensional grid. Also, sampling from arbitrary distributions in high-dimensional spaces is less accurate when p is saved in a grid, and this is required at every step of gradient descent.

Some important details regarding the implementation of the algorithm that produces the full dual distribution, as well as the implementation of the algorithm that focuses on the training points, are presented in the following section.

3.5 Computational and implementation details

The previous sections describe how to obtain the dual distribution in two cases: the case where it is possible to compute the expected values in Equation 3.31, so that the full density of the dual

Algorithm 1 Exact dual distribution

```
Input: $P_S$, $L(\cdot)$, learning rate $\eta$
Discretize domain $\mathcal{X} \to \mathcal{X}_D$
Initialize $p(x_n)$ for $x_n \in \mathcal{X}_D$
repeat
  for all $x_n \in \mathcal{X}_D$ do
    $\nabla(\mathcal{L}(p(x_n))) := \left(\mathbb{E}_{\substack{x_i \sim P \\ i \neq n}}[L(x_1, \ldots, x_N)] - \mathbb{E}_{\substack{x_i \sim P \\ i=1,\ldots,x_N}}[L(x_1, \ldots, x_N)]\right) p(x_n)$
    $p(x_n) := p(x_n) - \eta \nabla_\psi(\mathcal{L}(p(x_n)))$
    if $p(x_n) < 0$ then $p(x_n) = 0$
  end for
  Normalize $p$
until $(\nabla(\mathcal{L}(p)) = 0)$
```

distribution can be found, and the practical supervised learning case, where an approximation of the dual distribution is found at the training points. Some important implementation details of both algorithms are described in this section.

Algorithm 1 describes the procedure to obtain the dual distribution in the exact case, that is, when we are allowed to sample data from a desired distribution. This algorithm is the one used to obtain the dual distributions in the examples of Section 3.2.2.

We first discuss a significant speed up that can be applied in this case. In order to obtain the full function numerically, it is necessary to discretize the domain, and obtain the distribution at the desired resolution. Assume the chosen resolution is δ, then the number of times that $\mathbb{E}_{x_i \sim P, i \neq n}[L(x_1, \ldots, x_N)]$ must be computed is proportional to $(1/\delta)^d$, at each step of gradient descent, in d dimensions. This value is computed via MC simulation, and hence it can be a very computationally expensive operation.

When we use a squared loss function and a linear model with non-linear transformations, there is a closed form solution for the loss, given in Equation 3.34. However, it is still necessary to find $\mathbb{E}_{x_i}[\Phi_{MM}^{-1}]$. This matrix must be found through MC simulation for each value of x_n in the grid, while randomizing the remaining $N-1$ points generated according to P. However, there can be a significant saving in computation if we apply the Sherman-Morrison identity [60]:

$$\Phi_{MM}^{-1} = \Phi_{MM,n}^{-1} - \frac{\Phi_{MM,n}^{-1} \phi_M(x_n) \phi_M(x_n)^T \Phi_{MM,n}^{-1}}{1 + \phi_M(x_n)^T \Phi_{MM,n}^{-1} \phi_M(x_n)}, \tag{3.45}$$

where

$$\Phi_{MM,n} = \sum_{\substack{i=1 \\ i \neq n}}^{N} \phi_M(x_i) \phi_M(x_i)^T. \tag{3.46}$$

Hence, $\mathbb{E}_{x_i}[\Phi_{MM,n}^{-1}]$ can be computed once via MC simulation, and the value of $\mathbb{E}_{x_i}[\Phi_{MM}^{-1}]$ can be approximated using the identity, and substituting for $\Phi_{MM,n}^{-1}$ by its expected value. This allows us to

Algorithm 2 Approximate dual distribution for supervised learning

```
**Input:** $P_S$, $p_R$, $R = \{x_i\}_{i=1}^{N}$, $L(\cdot)$, learning rate $\eta$
Initialize $p(x_i)$ for $x_i \in R$
**repeat**
  $w := p./p_R$ (element-wise division)
  **for** all $x_i \in R$ **do**
    $w' := w$
    $\mathbb{E}_{\substack{x_i \sim P \\ i=1,\ldots,x_N}}[L(x_1,\ldots,x_N)] := L(w; x_1,\ldots,x_N)$
    $w'(x_i) := w(x_i) + 1$
    Normalize $w'$ so that $\sum_i w'_i = N$
    $\mathbb{E}_{\substack{x_i \sim P \\ i \neq n}}[L(x_1,\ldots,x_N)] := L(w'; x_1,\ldots,x_N)$
    $\nabla(\mathcal{L}(p(x_n)) := \left(\mathbb{E}_{\substack{x_i \sim P \\ i \neq n}}[L(x_1,\ldots,x_N)] - \mathbb{E}_{\substack{x_i \sim P \\ i=1,\ldots,x_N}}[L(x_1,\ldots,x_N)]\right) p(x_n)$
    $p(x_n) := p(x_n) - \eta \nabla_\psi(\mathcal{L}(p(x_n))$
    **if** $p(x_n) < 0$ **then** $p(x_n) = 0$
  **end for**
  Normalize $p$
**until** $(\nabla(\mathcal{L}(p)) = 0)$
```

compute, with a single MC simulation, the value of $\mathbb{E}_{x_i \sim P, i \neq n}[L(x_1,\ldots,x_N)]$ at every desired x_n.

Another computational consideration that should be taken into account regards the constraint satisfaction at each step. Although the update at each step, given by Equation 3.33, guarantees that p integrates to 1 if the initial p is a proper pdf, numerically there might be small errors that can make the resulting density add up to a value slightly different from 1. Hence, in the implementation we normalize p at each step to avoid instability issues.

We also noticed that carrying out the minimization in the p space rather than in the ψ space was much quicker and usually led to solutions that yield the lowest out-of-sample error. The only drawback is that the positivity constraint must be also forced at each step. We did this by using the heuristic of setting to zero at each step any values that become negative.

Finally, for all experiments, p was initialized to be a uniform distribution in the finite domain. Another possible initialization point is $p = p_S$. If an alternative initialization is used, p must be a smooth function, and $p(x) > 0$ at every x. Otherwise, since the updates are proportional to $p(x)$, the points initialized at zero will not change throughout the descent.

Algorithm 2 describes the procedure to obtain an approximate dual distribution, in the supervised learning setting. The same considerations regarding the constraints of the minimization problem are taken into account.

3.6 Differences with active learning

The concept of a dual distribution in supervised learning is somewhat related to similar ideas in active learning and experimental design. Especially, the methods of 'batch' active learning, where a 'design' distribution is found in order to minimize the error, seems to be solving a similar problem to the dual distribution. However, the fundamental difference is that active learning finds such optimal distribution *given a particular target function.* Hence, most methods rely on the information given by the target function in order to find a better training distribution. A common example is when distributions give more weight to points around the boundaries of the target function. Yet, the problem of finding the dual distribution is *independent* of the specific target function. The Monte Carlo simulations presented in Chapter 2, as well as the bounds shown, average over different realizations of target functions.

For example, [43] describes an algorithm to find an appropriate 'design' distribution that will lower the out-of-sample error. In the algorithm proposed, a first parameter is estimated with s data points, and with this parameter the optimal design distribution is found. Having a new design distribution, $T - s$ points are sampled from it and a final parameter is then estimated. Notice, however, that the optimal design distribution is dependent on the target function. In the results we present, if a dual distribution is found given a particular test distribution, such distribution is optimal independently of the specific target function.

Other papers in the active learning community that focus on linear regression, like [63], seem closely related to our work. In the mentioned paper, the results apply to linear regression only, and consider the out-of-sample error conditioned on a given training set. The nice property of the out-of-sample error in linear regression is that it is independent of the target function. This is the reason why even in the active learning setting, the dependence of the target function disappears in this case and the mathematical analysis looks similar to the one we presented in Section 2.2. Yet, even though our analysis in Chapter 2 is done with linear regression and hence uses similar mathematical formulas, our approach is based on averaging over realizations of training sets and of targets functions in the supervised learning scenario, rather than in the cases addressed in the mentioned papers. Furthermore, the problem of finding the dual distribution and the results presented can be applied to other learning algorithms besides linear regression, both for classification and regression problems in the supervised learning setting as shown in the previous section.

Another difference that may stand out to the reader is the way the 'design' distribution is used once it is found in the active learning papers, as opposed to how we propose to use the dual distribution here. In the active learning scenario, points are sampled from the design distribution, but in order to avoid obtaining a biased estimator, as shown in [61], the loss function is weighted for these points with

$w(x) = q(x)/p(x)$, following their notation, where $q(x)$ is the test distribution $(P_S(x))$ and $p(x)$ is the 'design' distribution found. Notice that in the results presented in the simulations of Section 2.1 and in Tables 3.1 and 3.2, we do not re-weight the points but instead explicitly allow a mismatch between P_S and P_R. Furthermore, in the supervised learning setting, where the training set is fixed and we are not allowed to sample new points, we propose that matching algorithms, as the ones described in Section 1.2, be used to match the given training set to the dual distribution. In this case, the objective is to have weights $w(x) = p^\star_R(x)/p_S(x)$, so that the training set appears distributed as the dual distribution. These weights are actually inverse to those used in the active learning algorithms described. Although we are aware that the estimator computed in the linear regression setting will be biased when we use the dual distribution, we are concerned with minimizing the out-of-sample error, which takes into account both bias and variance, and hence we may obtain a biased estimator but improve the mean-squared error performance as the results show in Tables 3.1 and 3.2.

Furthermore, the results shown in [61] hold only in the asymptotic case, and since we are dealing with the supervised learning scenario where only a finite training sample is available, the same assumptions are not valid. Thus, it is no longer optimal to use the mentioned weighting mechanism when N is not sufficiently large, as also shown in [61]. In the active learning setting, it is desirable that as more points are sampled, the proposed algorithms have performance guarantees. Hence, the algorithms are designed to satisfy conditions such as consistency of the estimator, unbiasedness, etc., in the asymptotic case, which explains why the active learning algorithms use the above-mentioned weighting mechanism. In our setting, minimizing the out-of-sample performance with a fixed-size training set is our main objective, which is why the two approaches differ. As it is clear, the dual distribution serves a different purpose in the supervised learning setting than that of the active learning algorithms.

Having answered the first fundamental question posed in Chapter 1, is it better to have $P_R = P_S$, and having concluded that in fact P^*_R is the optimal training distribution to generate the dataset for training the learning algorithm, we move on to answer the second question. Is it advantageous to use weights to make the training set look like $P^\star_R$? We answer this question in the following chapter.