EE2211 : Introduction to Machine Learning (Fall 2020)
Ridge Regression, Linear Classification and Polynomial Regression

Vincent Y. F. Tan

In this brief document, we will cover several topics. First, we reinforce some concepts in ridge regression. We motivate the use of this form of regularization in the context of least squares estimation. We will discuss both the primal and dual forms. The latter will allow us to segue to the (optional) topic of kernels. Second, we discuss how linear models can be used for classification. Finally, we consider a certain form of generalized linear models for regression known as polynomial regression.

1 Motivation for Ridge Regression

In the previous lectures, you learned about least squares estimation. Let us recap that. We assume here that we have more data points m than dimensions or attributes d. The data or training samples are denoted, as usual, as row vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \in \mathbb{R}^{1\times d}$. For the sake of notational brevity, we omit the offset from our training samples. The corresponding target variables are denoted as $y_1, y_2, \ldots, y_m \in \mathbb{R}$. The data samples and the target variables are stacked in the *design matrix* and *target vector* as follows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix} \in \mathbb{R}^{m\times d} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \in \mathbb{R}^m. \tag{1}$$

We aim to find a good *weight* or *coefficient vector* $\mathbf{w} \in \mathbb{R}^d$ such that $\mathbf{X}\mathbf{w}$ is "close to" $\mathbf{y}$. Note that the i^{th} row of $\mathbf{X}\mathbf{w}$, i.e., $\sum_{j=1}^d x_{i,j}w_j$ is the *prediction* of the target of the i^{th} data point. This error in predicting the i^{th} data point is denoted as

$$e_i := y_i - \sum_{j=1}^d x_{i,j}w_j \qquad 1 \le i \le m. \tag{2}$$

To ensure that all errors are small—or equivalently, $\mathbf{X}\mathbf{w}$ is close to $\mathbf{y}$—we seek to minimize the sum of squares of the e_i's over $\mathbf{w}$, i.e.,

$$\text{minimize} \quad \sum_{i=1}^m e_i^2. \tag{3}$$

It is clear that this is equivalent to

$$\text{minimize} \sum_{i=1}^m \left(y_i - \sum_{j=1}^d x_{i,j}w_j \right)^2 \iff \text{minimize} \sum_{i=1}^m \Big(y_i - (\mathbf{X}\mathbf{w})_i \Big)^2 \iff \text{minimize} \ \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2. \tag{4}$$

We saw that if $\mathbf{X}$ has full column rank, the solution for $\mathbf{w}$ is the least squares solution, denoted as

$$\hat{\mathbf{w}}_{\text{ls}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}. \tag{5}$$

While this is elegant, there is substantial motivation to study refinements of the optimization problem in (3), or equivalently, (4). We state three reasons here.

1. Firstly, in modern data science applications, we often have data that is extremely high dimensional. In contrast, the number of data points is often small to moderate. Think of a clinical application where we have $m = 500$ subjects in our cohort. Getting a single person to be in a study is extremely costly. Indeed, we need to compensate the subject for her/his time. For each subject, however, we can collect a large amount of information using today's acquisition devices. Indeed, a single blood test can reveal a lot about a person's genomic information in the form of *single nucleotide polymorphisms* (SNPs). Furthermore, there are roughly 4 to 5 million SNPs in a person's genome and SNPs may give us information about whether a person is susceptible to certain diseases or ailments such as cancer. Thus d can be of the order of millions, which is certainly larger than the number of subjects in our study. As a result, $\mathbf{X}$ is now a wide matrix, which almost surely does not have full column rank and the solution (5) does not exist. While one can appeal to the least norm solution, there are better solutions that make explicit use of prior information about the solution.

2. Secondly, we often want to go beyond using linear models to do prediction because of the flexibility that such richer models affords us. We have seen from the lecture that we can use polynomial models (see Section 4) in which case the effective number of features is $\binom{p+d}{d}$ where p is the order of the polynomial (show this by looking up *monomial numbers*!). In fact, there is often motivation to use infinite-dimensional models, in which there are infinitely many features. In this case, the feature vectors can be *kernelized* using the so-called radial basis functions. Clearly, the new dimension (which may be even be infinite) exceeds that of the number of samples m and (5) is no longer applicable.

3. Finally, even if m is larger than d, we often want to *stabilize* and *robustify* the solution so that it *generalizes* well. By regularizing the optimization problems in (3) and (4), we often get more stable solutions that do not fluctuate as much under small (or even large) changes in the data. We will see this when we discuss about the *bias-variance tradeoff* in subsequent lectures.

2 Ridge Regression

The way we achieve this stability or robustness in the least squares solution is to consider a *regularized* version of the optimization problem in (4). We consider the *ridge regression* or *Tikhonov regularization* problem

$$\text{minimize} \quad \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2, \tag{6}$$

where $\lambda > 0$ is known as the *regularization parameter*. Let us try to understand this a bit further. The first part of the objective function $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$ is the same; it forces the solution $\mathbf{w}$ to be such that the predictions contained in the vector $\mathbf{X}\mathbf{w}$ are close to the targets contained in $\mathbf{y}$. This is the so-called *data fidelity* or *risk* term. The second term in (6)—also known as the *regularization term*—forces the solution $\mathbf{w}$ to be small. This has the effect of stabilizing or robustifying the solution as large values of $\mathbf{w}$ do not lead to favorable generalization on new test examples.

2.1 Primal Form

By differentiating the objective function in (6) with respect to $\mathbf{w}$, we see that the solution $\hat{\mathbf{w}}_\lambda$ satisfies

$$\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}_\lambda) - \lambda \hat{\mathbf{w}}_\lambda = \mathbf{0}. \tag{7}$$

In other words,

$$\hat{\mathbf{w}}_\lambda = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}. \tag{8}$$

This is the ridge regression solution in *primal form*. Note that the matrix $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ is always non-singular if $\lambda > 0$ because it is positive definite. (Indeed, $\mathbf{z}^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}) \mathbf{z} = \|\mathbf{X}\mathbf{z}\|^2 + \lambda \|\mathbf{z}\|^2 > 0$ for any $\mathbf{z} \neq \mathbf{0}$.)

We note two extreme cases. If $\lambda \to 0^+$, then $\hat{\mathbf{w}}_\lambda$ reduces to $\hat{\mathbf{w}}_{\text{ls}}$. In this case, we place our entire faith in the data $(\mathbf{X}, \mathbf{y})$ and omit the regularization term. If $\lambda \to \infty$, then $\hat{\mathbf{w}}_\lambda$ reduces to the zero vector. In this case, we do not trust the data at all and our predictions of any test example is 0 because $\mathbf{x}_{\text{new}}^\top \hat{\mathbf{w}}_0 = \mathbf{0}$ for any $\mathbf{x}_{\text{new}} \in \mathbb{R}^d$.

2.2 Dual Form

Let us examine the optimization problem (6) in greater detail. The objective function can be written as

$$\sum_{i=1}^{m}(y_i - \mathbf{x}_i^\top \mathbf{w})^2 + \lambda\|\mathbf{w}\|^2. \tag{9}$$

Differentiating this meticulously, we obtain that

$$-2\sum_{i=1}^{m}(y_i - \mathbf{x}_i^\top \mathbf{w})\mathbf{x}_i + 2\lambda\mathbf{w} = \mathbf{0}. \tag{10}$$

Rearranging, we see that the solution $\hat{\mathbf{w}}_\lambda$ satisfies

$$\hat{\mathbf{w}}_\lambda = \frac{1}{\lambda}\sum_{i=1}^{m}\left(y_i - \mathbf{x}_i^\top \hat{\mathbf{w}}_\lambda\right)\mathbf{x}_i. \tag{11}$$

For each $1 \le i \le m$, define the real number $a_i := \lambda^{-1}(y_i - \mathbf{x}_i^\top \hat{\mathbf{w}}_\lambda)$. Hence, $\hat{\mathbf{w}}_\lambda$ can be written as

$$\hat{\mathbf{w}}_\lambda = \sum_{i=1}^{m} a_i\mathbf{x}_i \in \operatorname{span}\big(\{\mathbf{x}_i\}_{i=1}^m\big). \tag{12}$$

While this is simple, the result in (12) is surprisingly deep and is a special instance of the *Representer Theorem* in statistical learning theory; see the seminal works by Kimeldorf and Wahba [KW70, Lemma 2.2] and Schölkopf, Herbrich and Smola [SHS01]. It says that no matter how high dimensional the data samples $\mathbf{x}_i$ are—they may even be infinite dimensional—we do not need to solve the high-dimensional optimization problem in (6) to find the optimal $\mathbf{w}$. All that is required to solve for $\hat{\mathbf{w}}_\lambda$ within a ridge regression framework is the set of m coefficients $\{a_i\}_{i=1}^m$! So we have reduced a d-dimensional optimization problem to one that is "only" m dimensional and m may be much smaller than d. Here, we note the importance of λ being positive. If $\lambda = 0$ (no regularization), this will no longer be true because (11) would not hold.

Now we notice from (12) that $\hat{\mathbf{w}}_\lambda$ can be written as

$$\hat{\mathbf{w}}_\lambda = \mathbf{X}^\top \mathbf{a} \tag{13}$$

where $\mathbf{a} = (a_1, a_2, \ldots, a_m)^\top \in \mathbb{R}^m$ is the vector of coefficients that defines the optimal ridge regularized weight vector $\hat{\mathbf{w}}_\lambda$. In other words, $\hat{\mathbf{w}}_\lambda$ is in the the column space (range) of $\mathbf{X}^\top$. Substituting this into (6), we obtain the optimization problem (in $\mathbf{a}$)

$$\begin{aligned} & \text{minimize} \quad \left\|\mathbf{y} - \mathbf{X}\mathbf{X}^\top\mathbf{a}\right\|^2 + \lambda\|\mathbf{X}^\top\mathbf{a}\|^2 \\ \iff \quad & \text{minimize} \quad \mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\mathbf{X}^\top\mathbf{a} + \mathbf{a}^\top(\mathbf{X}\mathbf{X}^\top)^\top(\mathbf{X}\mathbf{X}^\top)\mathbf{a} + \lambda\mathbf{a}^\top\mathbf{X}\mathbf{X}^\top\mathbf{a}. \end{aligned} \tag{14}$$

Differentiating and solving for $\mathbf{a}$ yields

$$\begin{aligned} \mathbf{X}\mathbf{X}^\top\mathbf{y} &= (\mathbf{X}\mathbf{X}^\top)^\top(\mathbf{X}\mathbf{X}^\top)\mathbf{a} + \lambda\mathbf{X}\mathbf{X}^\top\mathbf{a} && (15)\\ &= (\mathbf{X}\mathbf{X}^\top)(\mathbf{X}\mathbf{X}^\top)\mathbf{a} + \lambda(\mathbf{X}\mathbf{X}^\top)\mathbf{a} && (16)\\ &= (\mathbf{X}\mathbf{X}^\top)\big(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}\big)\mathbf{a}. && (17) \end{aligned}$$

We note that (16) holds because $\mathbf{X}\mathbf{X}^\top$ is symmetric. The set of $\mathbf{a}$ that satisfies (18) is not a singleton if $\mathbf{X}$ does not have full row rank. In fact, the set of solutions $\mathbf{a}$ that satisfies (17) is

$$\Big\{\big(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}\big)^{-1}\mathbf{y} + \mathbf{z} : \mathbf{z} \in \mathcal{N}(\mathbf{X}\mathbf{X}^\top)\Big\}. \tag{18}$$

However, we note from a simple argument[1] that $\mathcal{N}(\mathbf{X}\mathbf{X}^\top) = \mathcal{N}(\mathbf{X}^\top)$ and so substituting (18) into (13), we see that the *dual form* of the solution is

$$\hat{\mathbf{w}}_\lambda = \mathbf{X}^\top \mathbf{a} = \mathbf{X}^\top \Big(\big(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}\big)^{-1} \mathbf{y} + \mathbf{z} \Big) = \mathbf{X}^\top \big(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}\big)^{-1} \mathbf{y}, \tag{19}$$

since $\mathbf{X}^\top \mathbf{z} = \mathbf{0}$. Even though $\mathbf{a}$ may not be unique, $\hat{\mathbf{w}}_\lambda$ is.

Note that the dual solution in (19) is really cool. Originally in the primal form of the solution in (8), we needed to invert a $d \times d$ matrix $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_d$. This costs $O(d^3)$ operations (additions and multiplications) in general. As mentioned, d could be really large and could even be infinite. In the dual form, however, we (only) need to invert an $m \times m$ matrix $\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_m$. This costs $O(m^3)$, which is much less than $O(d^3)$! We note that m is always finite because we always only have finite number of training samples.

In Appendix A.1, we use the Woodbury matrix identity to show that the primal and dual forms of the least squares solution are equal for any $\lambda > 0$.

2.3 Interpretation In Terms of Kernels (Advanced and Optional)

Substituting $\sum_{i=1}^m a_i \mathbf{x}_i$ for $\mathbf{w}$ in (9), we see that the optimization over $\mathbf{w}$ is equivalent to optimizing the following function over the vector $(a_1, a_2, \ldots, a_m) \in \mathbb{R}^m$:

$$J(a_1, a_2, \ldots, a_m) = \sum_{i=1}^m \left(y_i - \Big(\sum_{j=1}^m a_i \mathbf{x}_j \Big)^\top \mathbf{x}_i \right)^2 + \lambda \left\| \sum_{j=1}^m a_i \mathbf{x}_j \right\|^2 \tag{20}$$

$$= \sum_{i=1}^m \left(y_i - \sum_{j=1}^m a_i \mathbf{x}_j^\top \mathbf{x}_i \right)^2 + \lambda \sum_{i=1}^m \sum_{j=1}^m a_i a_j \mathbf{x}_i^\top \mathbf{x}_j \tag{21}$$

$$= \sum_{i=1}^m \left(y_i - \sum_{j=1}^m a_i K(\mathbf{x}_i, \mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^m \sum_{j=1}^m a_i a_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{22}$$

where $K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ is the inner product between two vectors. This says that even if d is much larger than m, we simply have to solve a lower-dimensional optimization problem of m dimensions—the m components in $\mathbf{a}$—to obtain $\mathbf{w}$ in the form in (12). In fact, $\mathbf{a}$ is given exactly in (18). Furthermore, all that is needed in this endeavour are the pairwise inner products between the feature vectors $\mathbf{K} = [K(\mathbf{x}_i, \mathbf{x}_j)]$, a matrix of size $m \times m$. There is no need to "look at" the feature vectors; all we need are pairwise similarities between them. So we do not even need the $\mathbf{x}_i$'s to be in $\mathbb{R}^d$; they could be text or string data [LSST+02] or other categorical variables. There is a lot more we can say here, but we need to introduce the notion of *kernels* [SSB18], which unfortunately is beyond the scope of the course.

3 Linear Models for Classification

Recall that in classification, the label y_i belongs to a finite set. We now briefly discuss how to extend the preceding framework into the realm of classification.

3.1 Binary Classification

In the simplest form of classification, one assumes that each y_i can only take on two values; this is called *binary classification*. The two values can be encoded in any reasonable way, but for the sake of concreteness,

[1] The only non-trivial direction is $\mathcal{N}(\mathbf{X}\mathbf{X}^\top) \subset \mathcal{N}(\mathbf{X}^\top)$. Take $\mathbf{z} \in \mathcal{N}(\mathbf{X}\mathbf{X}^\top)$. Then $\mathbf{X}\mathbf{X}^\top \mathbf{z} = \mathbf{0}$. Pre-multiply by $\mathbf{z}^\top$ to obtain $\mathbf{z}^\top \mathbf{X}\mathbf{X}^\top \mathbf{z} = 0$. Hence, $\|\mathbf{X}^\top \mathbf{z}\|^2 = 0$ and so $\mathbf{X}^\top \mathbf{z} = \mathbf{0}$ or equivalently $\mathbf{z} \in \mathcal{N}(\mathbf{X}^\top)$.

clarity, and simplicity, we assume that $y_i \in \{-1, +1\}$. The samples in $\mathcal{P} = \{\mathbf{x}_i : y_i = +1\}$ are from the *positive class* and the samples $\mathcal{N} := \{\mathbf{x}_i : y_i = -1\}$ are from the *negative class*.

Given what we have learned about linear regression thus far, the binary classification formulation and algorithm that we propose here is conceptually rather straightforward. The dataset $\mathcal{D} := \{(\mathbf{x}_i, y_i) : 1 \leq i \leq m\}$ is available to us where now $y_i \in \{-1, +1\}$. If we decide to use an offset (i.e., a bias term), then similarly to (1), we form the design matrix and target vector, except that we note that the target or label vector now lives in $\{-1, +1\}^m$. The same procedures—least squares, regularized least squares (ridge regression) in both the primal and dual forms—can be used to learn a weight vector $\mathbf{w} \in \mathbb{R}^{d'}$ where $d' = d$ if no offset is used or $d' = d + 1$ if an offset term is included into the model. Call the learned weight vector $\hat{\mathbf{w}}$.

Let us assume that there is no offset. For prediction purposes, we are given a new example $\mathbf{x}_{\text{new}} \in \mathbb{R}^d$ and would like to predict its class, i.e., whether it belongs to the positive or negative class. Our prediction is

$$\hat{y}_{\text{new}} = \text{sgn}\big(\mathbf{x}_{\text{new}}^\top \hat{\mathbf{w}}\big) \tag{23}$$

where sgn returns 1 of the argument is non-negative and -1 otherwise.[2]

While the formula in (23) is relatively simple, the interpretation is more important for the purposes of generalizing to the multiclass case in Section 3.2. What the rule in (23) is telling us is that we should declare the class of $\mathbf{x}_{\text{new}}$ to be positive (resp. negative) if its inner product with $\hat{\mathbf{w}}$ is positive (resp. negative), i.e., the angle it makes with $\hat{\mathbf{w}}$ is acute (resp. obtuse). Roughly speaking, this means that $\mathbf{x}_{\text{new}}$ is "more similar" (resp. "more dissimilar") to the samples in the positive class $\mathcal{P}$ (resp. the negative class $\mathcal{N}$).

3.2 Multiclass Classification

In real-world applications, y_i may take on many possibilities. For instance, the image classification example we discussed in Tutorial 1 posits that each image contains one out of a possible $c = 10$ animals.[3] Thus, we must expand the scope of our discussion for linear classification. We perform one-hot encoding of the class label and this is stored in a *label matrix* $\mathbf{Y} = [y_{i,k}] \in \{0, 1\}^{m \times c}$, where $y_{i,k} = 1$ means that training example $i \in \{1, 2, \ldots, m\}$ belongs to class $k \in \{1, 2, \ldots, c\}$. Note that each row of $\mathbf{Y}$ contains only one 1; the rest of the entries are 0. This is because we assume that each training example belongs to *exactly* one class.

In this framework, we aim to find $\mathbf{W} \in \mathbb{R}^{d \times c}$ such that $\mathbf{XW} \approx \mathbf{Y}$. To quantify the approximation error, we generalize (4) and minimize the sum of squares of all the entries in the error matrix $\mathbf{E} := \mathbf{Y} - \mathbf{XW}$; this is known as the square of the *Frobenius norm* of $\mathbf{E}$. That is, we consider the optimization problem

$$\text{minimize} \quad \|\mathbf{Y} - \mathbf{XW}\|_{\text{F}}^2 = \sum_{i=1}^{m} \sum_{k=1}^{c} \left(y_{ik} - \sum_{j=1}^{d} x_{i,j} w_{j,k} \right)^2 . \tag{24}$$

Going through the same procedure to optimize over the matrix $\mathbf{W}$, we find that its least squares solution is

$$\hat{\mathbf{W}}_{\text{ls}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} \in \mathbb{R}^{d \times c}, \tag{25}$$

assuming again that $\mathbf{X}$ has full column rank. Given a new test sample $\mathbf{x}_{\text{new}} \in \mathbb{R}^d$, analogously to the rule for binary classification in (23), we declare its class to be

$$\hat{y}_{\text{new}} = \underset{k \in \{1,2,\ldots,c\}}{\arg\max} \; \mathbf{x}_{\text{new}}^\top \hat{\mathbf{W}}_{\text{ls}}(:, k). \tag{26}$$

That is, we find the label $k \in \{1, 2, \ldots, c\}$ that maximizes the "correlation" or "similarity" between the new test sample $\mathbf{x}_{\text{new}}$ and the weights corresponding to that class, i.e., the k^{th} column of $\hat{\mathbf{W}}_{\text{ls}}(:, k)$.

[2] The definition of sgn(0) is immaterial in most engineering applications.

[3] The well known ImageNet dataset contains $m \approx 14 \times 10^6$ images and $c \approx 2 \times 10^4$ labels or categories.

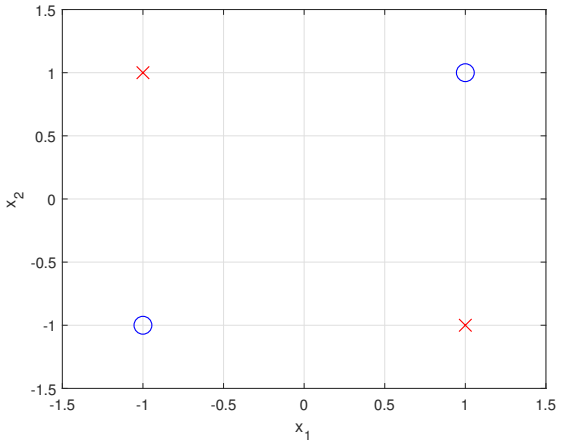

Figure 1: XOR example which is not linearly separable

4 Polynomial Regression

Up till now, our discussion has focused on *linear* models. However, there is significant motivation to go beyond linear models. Consider the dataset in Fig. 1, which is labelled as follows:

$$\begin{aligned} \mathbf{x}_1 &= \begin{bmatrix} +1 & +1 \end{bmatrix} \qquad y_1 = +1 \\ \mathbf{x}_2 &= \begin{bmatrix} -1 & +1 \end{bmatrix} \qquad y_2 = -1 \\ \mathbf{x}_3 &= \begin{bmatrix} +1 & -1 \end{bmatrix} \qquad y_3 = -1 \\ \mathbf{x}_4 &= \begin{bmatrix} -1 & -1 \end{bmatrix} \qquad y_4 = +1. \end{aligned} \tag{27}$$

For obvious reasons, this is known as the XOR problem. No matter how hard we try, we *cannot* find a weight vector (with offset term) $\mathbf{w} = (w_0, w_1, w_2)^\top \in \mathbb{R}^3$ such that all the training samples are classified correctly, i.e., $y_i = \text{sgn}([1, \mathbf{x}_i]^\top \mathbf{w})$ for all $1 \leq i \leq 4$. This dataset is *not linearly separable*. However, notice that there is a simple nonlinear classifier that does the job. If we take the products of the first and second components of each feature vector, we obtain

$$x_{11}x_{12} = +1, \quad x_{21}x_{22} = -1, \quad x_{31}x_{32} = -1, \quad x_{41}x_{42} = +1. \tag{28}$$

Voila! Now we have transformed our dataset into one that is linearly separable because I can simply choose w_{12}, the weight associated to the product of the first and second components to be 1. Then it is easy to check that $\text{sgn}(w_{12}x_{i1}x_{i2}) = y_i$ for all $1 \leq i \leq 4$. We have achieved zero training error.

This simple observation motivates the use of products (or, in general, nonlinear functions) of individual features. Suppose our feature vectors are one-dimensional, i.e., $x_i \in \mathbb{R}$ for all $1 \leq i \leq m$. Then, for regression and similarly for classification, we might want to consider going beyond the linear model $y = w_0 + w_1 x$ to learn a richer model involving powers of the x, i.e.,

$$y = w_0 + w_1 x + w_2 x^2 + \ldots + w_p x^p. \tag{29}$$

This is known as a *degree-p (univariate) polynomial* as the maximum power of the sole indeterminate x is p. In machine learning however, we have many features in most datasets. For the sake of discussion, say we have two features—x_1 and x_2. In this case, we have to introduce the notion of a *degree-p (bivariate and, in general, multivariate) monomial* which is a term of the form $x_1^a x_2^b$ where a and b are non-negative integers satisfying $a + b = p$. Hence, for two features x_1 and x_2, an *degree-2 (bivariate) polynomial* model is

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 + w_{11} x_1^2 + w_{22} x_2^2. \tag{30}$$

Here the vector of weights to be learned is $\mathbf{w} = [w_0, w_1, w_2, w_{12}, w_{11}, w_{22}]^\top$. Note that for the XOR problem a $\mathbf{w}$ that works in the sense of achieving zero training error is $\mathbf{w} = [0, 0, 0, 1, 0, 0]^\top$. In effect, what this $\mathbf{w}$ does it to partition the (x_1, x_2) plane as in Fig. 1 into two parts—the first and third quadrant to be declared as the region that contains the samples from the positive class and the second and fourth quadrant as the region that contains negatively labelled samples.

While this theory allows us to learn complicated (more precisely, nonlinear) decision boundaries, the number of weights grows rapidly with the degree of the polynomial p and the original number of features d. In the above example with $p = d = 2$, we had a total of 6 weights to be learned—$w_0, w_1, w_2, w_{12}, w_{11}, w_{22}$. In general, one can prove that the total number of weights is $\binom{p+d}{d}$. However, there is an easy way to ameliorate this explosive growth in the number of weights; one exploits a theory known as *kernels* (discussed briefly in Section 2.3). This is somewhat similar to the discussion leading to the dual solution in Section 2.2.

A.1 Equivalence of Primal and Dual Solutions (Optional)

Here, we show that for any $\lambda > 0$ and for any $(\mathbf{X}, \mathbf{y})$, we have

$$(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})^{-1} \mathbf{y}. \tag{31}$$

For this, we use the *Woodbury matrix identity*,

$$(\mathbf{I} + \mathbf{U}\mathbf{V})^{-1} = \mathbf{I} - \mathbf{U}(\mathbf{I} + \mathbf{V}\mathbf{U})^{-1}\mathbf{V}. \tag{32}$$

Starting from the expression on the right of (31), we have

$$\begin{aligned}
\mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})^{-1} \mathbf{y} &= \lambda^{-1} \mathbf{X}^\top (\mathbf{I} + \lambda^{-1} \mathbf{X}\mathbf{X}^\top)^{-1} \mathbf{y} && (33)\\
&= \lambda^{-1} \mathbf{X}^\top \left[\mathbf{I} - \lambda^{-1} \mathbf{X} (\mathbf{I} + \lambda^{-1} \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \right] \mathbf{y} && (34)\\
&= \lambda^{-1} (\mathbf{X}^\top \mathbf{y} - \mathbf{X}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}) && (35)\\
&= \lambda^{-1} (\mathbf{I} - \mathbf{X}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}) \mathbf{X}^\top \mathbf{y} && (36)\\
&= \lambda^{-1} \left[\mathbf{I} - (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} + \lambda \mathbf{I} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \right] \mathbf{X}^\top \mathbf{y} && (37)\\
&= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}, && (38)
\end{aligned}$$

where (34) follows from the Woodbury matrix identity with the identifications $\mathbf{U} \equiv \lambda^{-1}\mathbf{X}$ and $\mathbf{V} \equiv \mathbf{X}^\top$. This is an alternative proof, based solely on matrix algebra, that the primal and dual forms of the least squares solutions in (8) and (19) are the same.

References

[KW70] G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Ann. Math. Statist.*, 41:495–502, 1970.

[LSST+02] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, Feb 2002.

[SHS01] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *International Conference on Computational Learning Theory*, pages 416–426. Lecture Notes in Computer Science, 2001.

[SSB18] B. Schölkopf, A. J. Smola, and F. Bach. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2018.