9

Linear Predictors

In this chapter we will study the family of linear predictors, one of the most useful families of hypothesis classes. Many learning algorithms that are being widely used in practice rely on linear predictors, first and foremost because of the ability to learn them efficiently in many cases. In addition, linear predictors are intuitive, are easy to interpret, and fit the data reasonably well in many natural learning problems.

We will introduce several hypothesis classes belonging to this family – halfspaces, linear regression predictors, and logistic regression predictors – and present relevant learning algorithms: linear programming and the Perceptron algorithm for the class of halfspaces and the Least Squares algorithm for linear regression. This chapter is focused on learning linear predictors using the ERM approach; however, in later chapters we will see alternative paradigms for learning these hypothesis classes.

First, we define the class of affine functions as

$$L_d = \{h_{\mathbf{w},b} : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\},$$

where

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \left(\sum_{i=1}^{d} w_i x_i \right) + b.$$

It will be convenient also to use the notation

$$L_d = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle + b : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\},$$

which reads as follows: L_d is a set of functions, where each function is parameterized by $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, and each such function takes as input a vector x and returns as output the scalar $\langle \mathbf{w}, \mathbf{x} \rangle + b$.

The different hypothesis classes of linear predictors are compositions of a function $\phi : \mathbb{R} \to \mathcal{Y}$ on L_d. For example, in binary classification, we can choose ϕ to be the sign function, and for regression problems, where $\mathcal{Y} = \mathbb{R}$, ϕ is simply the identity function.

It may be more convenient to incorporate b, called the *bias*, into $\mathbf{w}$ as an extra coordinate and add an extra coordinate with a value of 1 to all $\mathbf{x} \in \mathcal{X}$; namely,

let $\mathbf{w}' = (b, w_1, w_2, \ldots w_d) \in \mathbb{R}^{d+1}$ and let $\mathbf{x}' = (1, x_1, x_2, \ldots, x_d) \in \mathbb{R}^{d+1}$. Therefore,

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \langle \mathbf{w}', \mathbf{x}' \rangle.$$

It follows that each affine function in $\mathbb{R}^d$ can be rewritten as a homogenous linear function in $\mathbb{R}^{d+1}$ applied over the transformation that appends the constant 1 to each input vector. Therefore, whenever it simplifies the presentation, we will omit the bias term and refer to L_d as the class of homogenous linear functions of the form $h_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$.

Throughout the book we often use the general term "linear functions" for both affine functions and (homogenous) linear functions.

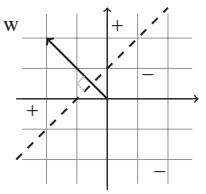
9.1 HALFSPACES

The first hypothesis class we consider is the class of halfspaces, designed for binary classification problems, namely, $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$. The class of halfspaces is defined as follows:

$$HS_d = \text{sign} \circ L_d = \{\mathbf{x} \mapsto \text{sign}(h_{\mathbf{w},b}(\mathbf{x})) : h_{\mathbf{w},b} \in L_d\}.$$

In other words, each halfspace hypothesis in HS_d is parameterized by $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ and upon receiving a vector $\mathbf{x}$ the hypothesis returns the label $\text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$.

To illustrate this hypothesis class geometrically, it is instructive to consider the case $d = 2$. Each hypothesis forms a hyperplane that is perpendicular to the vector $\mathbf{w}$ and intersects the vertical axis at the point $(0, -b/w_2)$. The instances that are "above" the hyperplane, that is, share an acute angle with $\mathbf{w}$, are labeled positively. Instances that are "below" the hyperplane, that is, share an obtuse angle with $\mathbf{w}$, are labeled negatively.



In Section 9.1.3 we will show that $\text{VCdim}(HS_d) = d + 1$. It follows that we can learn halfspaces using the ERM paradigm, as long as the sample size is $\Omega\left(\frac{d+\log(1/\delta)}{\epsilon}\right)$. Therefore, we now discuss how to implement an ERM procedure for halfspaces.

We introduce in the following two solutions to finding an ERM halfspace in the realizable case. In the context of halfspaces, the realizable case is often referred to as the "separable" case, since it is possible to separate with a hyperplane all the positive examples from all the negative examples. Implementing the ERM rule in the nonseparable case (i.e., the agnostic case) is known to be computationally hard (Ben-David and Simon, 2001). There are several approaches to learning nonseparable data. The most popular one is to use *surrogate loss functions*, namely, to learn a halfspace that does not necessarily minimize the empirical risk with the 0−1 loss, but rather with respect to a diffferent loss function. For example, in Section 9.3 we

will describe the logistic regression approach, which can be implemented efficiently even in the nonseparable case. We will study surrogate loss functions in more detail later on in Chapter 12.

9.1.1 Linear Programming for the Class of Halfspaces

Linear programs (LP) are problems that can be expressed as maximizing a linear function subject to linear inequalities. That is,

$$\begin{aligned} \max_{\mathbf{w}\in\mathbb{R}^d} \quad & \langle \mathbf{u}, \mathbf{w} \rangle \\ \text{subject to} \quad & A\mathbf{w} \geq \mathbf{v} \end{aligned}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the vector of variables we wish to determine, A is an $m \times d$ matrix, and $\mathbf{v} \in \mathbb{R}^m, \mathbf{u} \in \mathbb{R}^d$ are vectors. Linear programs can be solved efficiently,[1] and furthermore, there are publicly available implementations of LP solvers.

We will show that the ERM problem for halfspaces in the realizable case can be expressed as a linear program. For simplicity, we assume the homogenous case. Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a training set of size m. Since we assume the realizable case, an ERM predictor should have zero errors on the training set. That is, we are looking for some vector $\mathbf{w} \in \mathbb{R}^d$ for which

$$\text{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle) = y_i, \qquad \forall i = 1, \ldots, m.$$

Equivalently, we are looking for some vector $\mathbf{w}$ for which

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0, \qquad \forall i = 1, \ldots, m.$$

Let $\mathbf{w}^*$ be a vector that satisfies this condition (it must exist since we assume realizability). Define $\gamma = \min_i (y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle)$ and let $\bar{\mathbf{w}} = \frac{\mathbf{w}^*}{\gamma}$. Therefore, for all i we have

$$y_i \langle \bar{\mathbf{w}}, x_i \rangle = \frac{1}{\gamma} y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle \geq 1.$$

We have thus shown that there exists a vector that satisfies

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1, \qquad \forall i = 1, \ldots, m. \tag{9.1}$$

And clearly, such a vector is an ERM predictor.

To find a vector that satisfies Equation (9.1) we can rely on an LP solver as follows. Set A to be the $m \times d$ matrix whose rows are the instances multiplied by y_i. That is, $A_{i,j} = y_i x_{i,j}$, where $x_{i,j}$ is the j'th element of the vector $\mathbf{x}_i$. Let $\mathbf{v}$ be the vector $(1, \ldots, 1) \in \mathbb{R}^m$. Then, Equation (9.1) can be rewritten as

$$A\mathbf{w} \geq \mathbf{v}.$$

The LP form requires a maximization objective, yet all the $\mathbf{w}$ that satisfy the constraints are equal candidates as output hypotheses. Thus, we set a "dummy" objective, $\mathbf{u} = (0, \ldots, 0) \in \mathbb{R}^d$.

[1] Namely, in time polynomial in m, d, and in the representation size of real numbers.

9.1.2 Perceptron for Halfspaces

A different implementation of the ERM rule is the Perceptron algorithm of Rosenblatt (Rosenblatt 1958). The Perceptron is an iterative algorithm that constructs a sequence of vectors $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots$. Initially, $\mathbf{w}^{(1)}$ is set to be the all-zeros vector. At iteration t, the Perceptron finds an example i that is mislabeled by $\mathbf{w}^{(t)}$, namely, an example for which $\text{sign}(\langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle) \neq y_i$. Then, the Perceptron updates $\mathbf{w}^{(t)}$ by adding to it the instance $\mathbf{x}_i$ scaled by the label y_i. That is, $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + y_i \mathbf{x}_i$. Recall that our goal is to have $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0$ for all i and note that

$$y_i \langle \mathbf{w}^{(t+1)}, \mathbf{x}_i \rangle = y_i \langle \mathbf{w}^{(t)} + y_i \mathbf{x}_i, \mathbf{x}_i \rangle = y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle + \|\mathbf{x}_i\|^2.$$

Hence, the update of the Perceptron guides the solution to be "more correct" on the i'th example.

Batch Perceptron

input: A training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
initialize: $\mathbf{w}^{(1)} = (0, \ldots, 0)$
  for $t = 1, 2, \ldots$
    if $(\exists\, i$ s.t. $y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle \leq 0)$ then
      $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + y_i \mathbf{x}_i$
    else
      output $\mathbf{w}^{(t)}$

The following theorem guarantees that in the realizable case, the algorithm stops with all sample points correctly classified.

Theorem 9.1. *Assume that* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ *is separable, let* $B = \min\{\|\mathbf{w}\| : \forall i \in [m],\ y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1\}$, *and let* $R = \max_i \|\mathbf{x}_i\|$. *Then, the Perceptron algorithm stops after at most* $(RB)^2$ *iterations, and when it stops it holds that* $\forall i \in [m],\ y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle > 0$.

Proof. By the definition of the stopping condition, if the Perceptron stops it must have separated all the examples. We will show that if the Perceptron runs for T iterations, then we must have $T \leq (RB)^2$, which implies the Perceptron must stop after at most $(RB)^2$ iterations.

Let $\mathbf{w}^\star$ be a vector that achieves the minimum in the definition of B. That is, $y_i \langle \mathbf{w}^\star, x_i \rangle \geq 1$ for all i, and among all vectors that satisfy these constraints, $\mathbf{w}^\star$ is of minimal norm.

The idea of the proof is to show that after performing T iterations, the cosine of the angle between $\mathbf{w}^\star$ and $\mathbf{w}^{(T+1)}$ is at least $\frac{\sqrt{T}}{RB}$. That is, we will show that

$$\frac{\langle \mathbf{w}^\star, \mathbf{w}^{(T+1)} \rangle}{\|\mathbf{w}^\star\| \, \|\mathbf{w}^{(T+1)}\|} \geq \frac{\sqrt{T}}{RB}. \tag{9.2}$$

By the Cauchy-Schwartz inequality, the left-hand side of Equation (9.2) is at most 1. Therefore, Equation (9.2) would imply that

$$1 \geq \frac{\sqrt{T}}{RB} \quad \Rightarrow \quad T \leq (RB)^2,$$

which will conclude our proof.

To show that Equation (9.2) holds, we first show that $\langle \mathbf{w}^\star, \mathbf{w}^{(T+1)} \rangle \geq T$. Indeed, at the first iteration, $\mathbf{w}^{(1)} = (0, \ldots, 0)$ and therefore $\langle \mathbf{w}^\star, \mathbf{w}^{(1)} \rangle = 0$, while on iteration t, if we update using example $(\mathbf{x}_i, y_i)$ we have that

$$\begin{aligned}\langle \mathbf{w}^\star, \mathbf{w}^{(t+1)} \rangle - \langle \mathbf{w}^\star, \mathbf{w}^{(t)} \rangle &= \langle \mathbf{w}^\star, \mathbf{w}^{(t+1)} - \mathbf{w}^{(t)} \rangle \\ &= \langle \mathbf{w}^\star, y_i \mathbf{x}_i \rangle = y_i \langle \mathbf{w}^\star, \mathbf{x}_i \rangle \\ &\geq 1.\end{aligned}$$

Therefore, after performing T iterations, we get

$$\langle \mathbf{w}^\star, \mathbf{w}^{(T+1)} \rangle = \sum_{t=1}^{T} \left(\langle \mathbf{w}^\star, \mathbf{w}^{(t+1)} \rangle - \langle \mathbf{w}^\star, \mathbf{w}^{(t)} \rangle \right) \geq T, \tag{9.3}$$

as required.

Next, we upper bound $\|\mathbf{w}^{(T+1)}\|$. For each iteration t we have that

$$\begin{aligned}\|\mathbf{w}^{(t+1)}\|^2 &= \|\mathbf{w}^{(t)} + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}^{(t)}\|^2 + 2 y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle + y_i^2 \|\mathbf{x}_i\|^2 \\ &\leq \|\mathbf{w}^{(t)}\|^2 + R^2 \qquad (9.4)\end{aligned}$$

where the last inequality is due to the fact that example i is necessarily such that $y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle \leq 0$, and the norm of $\mathbf{x}_i$ is at most R. Now, since $\|\mathbf{w}^{(1)}\|^2 = 0$, if we use Equation (9.4) recursively for T iterations, we obtain that

$$\|\mathbf{w}^{(T+1)}\|^2 \leq TR^2 \;\Rightarrow\; \|\mathbf{w}^{(T+1)}\| \leq \sqrt{T} R. \tag{9.5}$$

Combining Equation (9.3) with Equation (9.5), and using the fact that $\|\mathbf{w}^\star\| = B$, we obtain that

$$\frac{\langle \mathbf{w}^{(T+1)}, \mathbf{w}^\star \rangle}{\|\mathbf{w}^\star\| \, \|\mathbf{w}^{(T+1)}\|} \geq \frac{T}{B\sqrt{T}R} = \frac{\sqrt{T}}{BR}.$$

We have thus shown that Equation (9.2) holds, and this concludes our proof. □

Remark 9.1. The Perceptron is simple to implement and is guaranteed to converge. However, the convergence rate depends on the parameter B, which in some situations might be exponentially large in d. In such cases, it would be better to implement the ERM problem by solving a linear program, as described in the previous section. Nevertheless, for many natural data sets, the size of B is not too large, and the Perceptron converges quite fast.

9.1.3 The VC Dimension of Halfspaces

To compute the VC dimension of halfspaces, we start with the homogenous case.

Theorem 9.2. *The VC dimension of the class of homogenous halfspaces in $\mathbb{R}^d$ is d.*

Proof. First, consider the set of vectors $\mathbf{e}_1, \ldots, \mathbf{e}_d$, where for every i the vector $\mathbf{e}_i$ is the all zeros vector except 1 in the i'th coordinate. This set is shattered by the class

of homogenous halfspaces. Indeed, for every labeling $y_1, \ldots, y_d$, set $\mathbf{w} = (y_1, \ldots, y_d)$, and then $\langle \mathbf{w}, \mathbf{e}_i \rangle = y_i$ for all i.

Next, let $\mathbf{x}_1, \ldots, \mathbf{x}_{d+1}$ be a set of $d+1$ vectors in $\mathbb{R}^d$. Then, there must exist real numbers $a_1, \ldots, a_{d+1}$, not all of them are zero, such that $\sum_{i=1}^{d+1} a_i \mathbf{x}_i = \mathbf{0}$. Let $I = \{i : a_i > 0\}$ and $J = \{j : a_j < 0\}$. Either I or J is nonempty. Let us first assume that both of them are nonempty. Then,

$$\sum_{i \in I} a_i \mathbf{x}_i = \sum_{j \in J} |a_j| \mathbf{x}_j.$$

Now, suppose that $\mathbf{x}_1, \ldots, \mathbf{x}_{d+1}$ are shattered by the class of homogenous classes. Then, there must exist a vector $\mathbf{w}$ such that $\langle \mathbf{w}, \mathbf{x}_i \rangle > 0$ for all $i \in I$ while $\langle \mathbf{w}, \mathbf{x}_j \rangle < 0$ for every $j \in J$. It follows that

$$0 < \sum_{i \in I} a_i \langle \mathbf{x}_i, \mathbf{w} \rangle = \left\langle \sum_{i \in I} a_i \mathbf{x}_i, \mathbf{w} \right\rangle = \left\langle \sum_{j \in J} |a_j| \mathbf{x}_j, \mathbf{w} \right\rangle = \sum_{j \in J} |a_j| \langle \mathbf{x}_j, \mathbf{w} \rangle < 0,$$

which leads to a contradiction. Finally, if J (respectively, I) is empty then the right-most (respectively, left-most) inequality should be replaced by an equality, which still leads to a contradiction. □

Theorem 9.3. *The VC dimension of the class of nonhomogenous halfspaces in $\mathbb{R}^d$ is $d+1$.*

Proof. First, as in the proof of Theorem 9.2, it is easy to verify that the set of vectors $\mathbf{0}, \mathbf{e}_1, \ldots, \mathbf{e}_d$ is shattered by the class of nonhomogenous halfspaces. Second, suppose that the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{d+2}$ are shattered by the class of nonhomogenous halfspaces. But, using the reduction we have shown in the beginning of this chapter, it follows that there are $d+2$ vectors in $\mathbb{R}^{d+1}$ that are shattered by the class of homogenous halfspaces. But this contradicts Theorem 9.2. □

9.2 LINEAR REGRESSION

Linear regression is a common statistical tool for modeling the relationship between some "explanatory" variables and some real valued outcome. Cast as a learning problem, the domain set $\mathcal{X}$ is a subset of $\mathbb{R}^d$, for some d, and the label set $\mathcal{Y}$ is the set of real numbers. We would like to learn a linear function $h : \mathbb{R}^d \to \mathbb{R}$ that best approximates the relationship between our variables (say, for example, predicting the weight of a baby as a function of her age and weight at birth). Figure 9.1 shows an example of a linear regression predictor for $d = 1$.

The hypothesis class of linear regression predictors is simply the set of linear functions,

$$\mathcal{H}_{reg} = L_d = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle + b : \mathbf{w} \in \mathbb{R}^d,\ b \in \mathbb{R}\}.$$

Next we need to define a loss function for regression. While in classification the definition of the loss is straightforward, as $\ell(h, (\mathbf{x}, y))$ simply indicates whether $h(\mathbf{x})$ correctly predicts y or not, in regression, if the baby's weight is 3 kg, both the predictions 3.00001 kg and 4 kg are "wrong," but we would clearly prefer the former over the latter. We therefore need to define how much we shall be "penalized" for

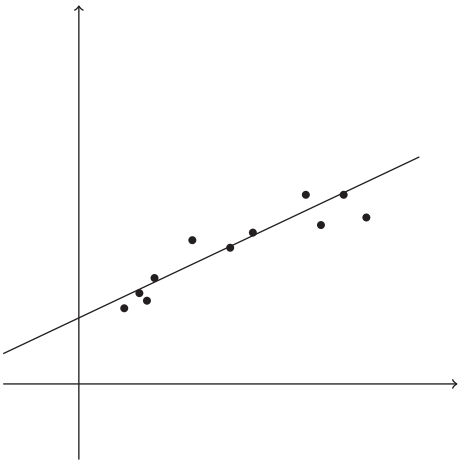


Figure 9.1. Linear regression for $d = 1$. For instance, the x-axis may denote the age of the baby, and the y-axis her weight.

the discrepancy between $h(\mathbf{x})$ and y. One common way is to use the squared-loss function, namely,

$$\ell(h, (\mathbf{x}, y)) = (h(\mathbf{x}) - y)^2.$$

For this loss function, the empirical risk function is called the Mean Squared Error, namely,

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} (h(\mathbf{x}_i) - y_i)^2.$$

In the next subsection, we will see how to implement the ERM rule for linear regression with respect to the squared loss. Of course, there are a variety of other loss functions that one can use, for example, the absolute value loss function, $\ell(h, (\mathbf{x}, y)) = |h(\mathbf{x}) - y|$. The ERM rule for the absolute value loss function can be implemented using linear programming (see Exercise 9.1).

Note that since linear regression is not a binary prediction task, we cannot analyze its sample complexity using the VC-dimension. One possible analysis of the sample complexity of linear regression is by relying on the "discretization trick" (see Remark 4.1 in Chapter 4); namely, if we are happy with a representation of each element of the vector $\mathbf{w}$ and the bias b using a finite number of bits (say a 64 bits floating point representation), then the hypothesis class becomes finite and its size is at most $2^{64(d+1)}$. We can now rely on sample complexity bounds for finite hypothesis classes as described in Chapter 4. Note, however, that to apply the sample complexity bounds from Chapter 4 we also need that the loss function will be bounded. Later in the book we will describe more rigorous means to analyze the sample complexity of regression problems.

9.2.1 Least Squares

Least squares is the algorithm that solves the ERM problem for the hypothesis class of linear regression predictors with respect to the squared loss. The ERM problem

with respect to this class, given a training set S, and using the homogenous version of L_d, is to find

$$\operatorname*{argmin}_{\mathbf{w}} L_S(h_{\mathbf{w}}) = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{m}\sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x}_i\rangle - y_i)^2.$$

To solve the problem we calculate the gradient of the objective function and compare it to zero. That is, we need to solve

$$\frac{2}{m}\sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x}_i\rangle - y_i)\mathbf{x}_i = 0.$$

We can rewrite the problem as the problem $A\mathbf{w} = \mathbf{b}$ where

$$A = \left(\sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^\top\right) \quad \text{and} \quad \mathbf{b} = \sum_{i=1}^{m} y_i \mathbf{x}_i. \tag{9.6}$$

Or, in matrix form:

$$A = \begin{pmatrix} \vdots & & \vdots \\ \mathbf{x}_1 & \dots & \mathbf{x}_m \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} \vdots & & \vdots \\ \mathbf{x}_1 & \dots & \mathbf{x}_m \\ \vdots & & \vdots \end{pmatrix}^\top, \tag{9.7}$$

$$\mathbf{b} = \begin{pmatrix} \vdots & & \vdots \\ \mathbf{x}_1 & \dots & \mathbf{x}_m \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}. \tag{9.8}$$

If A is invertible then the solution to the ERM problem is

$$\mathbf{w} = A^{-1}\mathbf{b}.$$

The case in which A is not invertible requires a few standard tools from linear algebra, which are available in Appendix C. It can be easily shown that if the training instances do not span the entire space of $\mathbb{R}^d$ then A is not invertible. Nevertheless, we can always find a solution to the system $A\mathbf{w} = \mathbf{b}$ because $\mathbf{b}$ is in the range of A. Indeed, since A is symmetric we can write it using its eigenvalue decomposition as $A = VDV^\top$, where D is a diagonal matrix and V is an orthonormal matrix (that is, $V^\top V$ is the identity $d \times d$ matrix). Define D^+ to be the diagonal matrix such that $D^+_{i,i} = 0$ if $D_{i,i} = 0$ and otherwise $D^+_{i,i} = 1/D_{i,i}$. Now, define

$$A^+ = VD^+V^\top \quad \text{and} \quad \hat{\mathbf{w}} = A^+\mathbf{b}.$$

Let $\mathbf{v}_i$ denote the i'th column of V. Then, we have

$$A\hat{\mathbf{w}} = AA^+\mathbf{b} = VDV^\top VD^+V^\top\mathbf{b} = VDD^+V^\top\mathbf{b} = \sum_{i:D_{i,i}\neq 0} \mathbf{v}_i\mathbf{v}_i^\top\mathbf{b}.$$

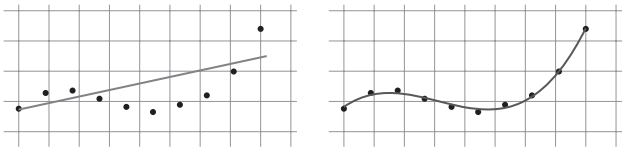
That is, $A\hat{\mathbf{w}}$ is the projection of $\mathbf{b}$ onto the span of those vectors $\mathbf{v}_i$ for which $D_{i,i} \neq 0$. Since the linear span of $\mathbf{x}_1, \dots, \mathbf{x}_m$ is the same as the linear span of those $\mathbf{v}_i$, and $\mathbf{b}$ is in the linear span of the $\mathbf{x}_i$, we obtain that $A\hat{\mathbf{w}} = \mathbf{b}$, which concludes our argument.

9.2.2 Linear Regression for Polynomial Regression Tasks

Some learning tasks call for nonlinear predictors, such as polynomial predictors. Take, for instance, a one dimensional polynomial function of degree n, that is,

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

where $(a_0, \ldots, a_n)$ is a vector of coefficients of size $n+1$. In the following we depict a training set that is better fitted using a 3rd degree polynomial predictor than using a linear predictor.



We will focus here on the class of one dimensional, n-degree, polynomial regression predictors, namely,

$$\mathcal{H}^n_{poly} = \{x \mapsto p(x)\},$$

where p is a one dimensional polynomial of degree n, parameterized by a vector of coefficients $(a_0, \ldots, a_n)$. Note that $\mathcal{X} = \mathbb{R}$, since this is a one dimensional polynomial, and $\mathcal{Y} = \mathbb{R}$, as this is a regression problem.

One way to learn this class is by reduction to the problem of linear regression, which we have already shown how to solve. To translate a polynomial regression problem to a linear regression problem, we define the mapping $\psi : \mathbb{R} \to \mathbb{R}^{n+1}$ such that $\psi(x) = (1, x, x^2, \ldots, x^n)$. Then we have that

$$p(\psi(x)) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = \langle \mathbf{a}, \psi(x) \rangle$$

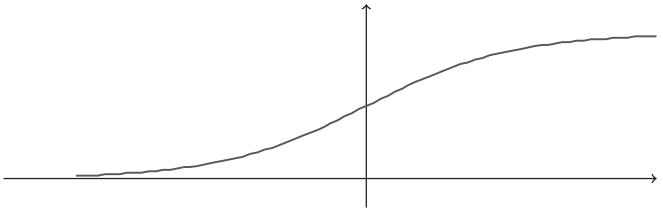
and we can find the optimal vector of coefficients $\mathbf{a}$ by using the Least Squares algorithm as shown earlier.

9.3 LOGISTIC REGRESSION

In logistic regression we learn a family of functions h from $\mathbb{R}^d$ to the interval $[0, 1]$. However, logistic regression is used for classification tasks: We can interpret $h(\mathbf{x})$ as the *probability* that the label of $\mathbf{x}$ is 1. The hypothesis class associated with logistic regression is the composition of a sigmoid function $\phi_{\text{sig}} : \mathbb{R} \to [0, 1]$ over the class of linear functions L_d. In particular, the sigmoid function used in logistic regression is the *logistic function*, defined as

$$\phi_{\text{sig}}(z) = \frac{1}{1 + \exp(-z)}. \tag{9.9}$$

The name "sigmoid" means "S-shaped," referring to the plot of this function, shown in the figure:



The hypothesis class is therefore (where for simplicity we are using homogenous linear functions):

$$H_{\text{sig}} = \phi_{\text{sig}} \circ L_d = \{\mathbf{x} \mapsto \phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle) : \mathbf{w} \in \mathbb{R}^d\}.$$

Note that when $\langle \mathbf{w}, \mathbf{x} \rangle$ is very large then $\phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle)$ is close to 1, whereas if $\langle \mathbf{w}, \mathbf{x} \rangle$ is very small then $\phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle)$ is close to 0. Recall that the prediction of the halfspace corresponding to a vector $\mathbf{w}$ is $\text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$. Therefore, the predictions of the halfspace hypothesis and the logistic hypothesis are very similar whenever $|\langle \mathbf{w}, \mathbf{x} \rangle|$ is large. However, when $|\langle \mathbf{w}, \mathbf{x} \rangle|$ is close to 0 we have that $\phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle) \approx \frac{1}{2}$. Intuitively, the logistic hypothesis is not sure about the value of the label so it guesses that the label is $\text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$ with probability slightly larger than 50%. In contrast, the halfspace hypothesis always outputs a deterministic prediction of either 1 or -1, even if $|\langle \mathbf{w}, \mathbf{x} \rangle|$ is very close to 0.

Next, we need to specify a loss function. That is, we should define how bad it is to predict some $h_{\mathbf{w}}(\mathbf{x}) \in [0, 1]$ given that the true label is $y \in \{\pm 1\}$. Clearly, we would like that $h_{\mathbf{w}}(\mathbf{x})$ would be large if $y = 1$ and that $1 - h_{\mathbf{w}}(\mathbf{x})$ (i.e., the probability of predicting -1) would be large if $y = -1$. Note that

$$1 - h_{\mathbf{w}}(\mathbf{x}) = 1 - \frac{1}{1 + \exp(-\langle \mathbf{w}, \mathbf{x} \rangle)} = \frac{\exp(-\langle \mathbf{w}, \mathbf{x} \rangle)}{1 + \exp(-\langle \mathbf{w}, \mathbf{x} \rangle)} = \frac{1}{1 + \exp(\langle \mathbf{w}, \mathbf{x} \rangle)}.$$

Therefore, any reasonable loss function would increase monotonically with $\frac{1}{1+\exp(y\langle \mathbf{w}, \mathbf{x} \rangle)}$, or equivalently, would increase monotonically with $1 + \exp(-y\langle \mathbf{w}, \mathbf{x} \rangle)$. The logistic loss function used in logistic regression penalizes $h_{\mathbf{w}}$ based on the log of $1 + \exp(-y\langle \mathbf{w}, \mathbf{x} \rangle)$ (recall that log is a monotonic function). That is,

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, y)) = \log\left(1 + \exp(-y\langle \mathbf{w}, \mathbf{x} \rangle)\right).$$

Therefore, given a training set $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, the ERM problem associated with logistic regression is

$$\operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 + \exp(-y_i \langle \mathbf{w}, \mathbf{x}_i \rangle)\right). \tag{9.10}$$

The advantage of the logistic loss function is that it is a *convex* function with respect to $\mathbf{w}$; hence the ERM problem can be solved efficiently using standard methods. We will study how to learn with convex functions, and in particular specify a simple algorithm for minimizing convex functions, in later chapters.

The ERM problem associated with logistic regression (Equation (9.10)) is identical to the problem of finding a Maximum Likelihood Estimator, a well-known

statistical approach for finding the parameters that maximize the joint probability of a given data set assuming a specific parametric probability function. We will study the Maximum Likelihood approach in Chapter 24.

9.4 SUMMARY

The family of linear predictors is one of the most useful families of hypothesis classes, and many learning algorithms that are being widely used in practice rely on linear predictors. We have shown efficient algorithms for learning linear predictors with respect to the zero-one loss in the separable case and with respect to the squared and logistic losses in the unrealizable case. In later chapters we will present the properties of the loss function that enable efficient learning.

Naturally, linear predictors are effective whenever we assume, as prior knowledge, that some linear predictor attains low risk with respect to the underlying distribution. In the next chapter we show how to construct nonlinear predictors by composing linear predictors on top of simple classes. This will enable us to employ linear predictors for a variety of prior knowledge assumptions.

9.5 BIBLIOGRAPHIC REMARKS

The Perceptron algorithm dates back to Rosenblatt (1958). The proof of its convergence rate is due to (Agmon 1954, Novikoff 1962). Least Squares regression goes back to Gauss (1795), Legendre (1805), and Adrain (1808).

9.6 EXERCISES

9.1 Show how to cast the ERM problem of linear regression with respect to the absolute value loss function, $\ell(h, (\mathbf{x}, y)) = |h(\mathbf{x}) - y|$, as a linear program; namely, show how to write the problem

$$\min_{\mathbf{w}} \sum_{i=1}^{m} |\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i|$$

as a linear program.
Hint: Start with proving that for any $c \in \mathbb{R}$,

$$|c| \;=\; \min_{a \geq 0} a \;\text{ s.t. }\; c \leq a \text{ and } c \geq -a.$$

9.2 Show that the matrix A defined in Equation (9.6) is invertible if and only if $\mathbf{x}_1, \ldots, \mathbf{x}_m$ span $\mathbb{R}^d$.

9.3 Show that Theorem 9.1 is tight in the following sense: For any positive integer m, there exist a vector $\mathbf{w}^* \in \mathbb{R}^d$ (for some appropriate d) and a sequence of examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ such that the following hold:

- $R = \max_i \|\mathbf{x}_i\| \leq 1$.
- $\|\mathbf{w}^*\|^2 = m$, and for all $i \leq m$, $y_i \langle \mathbf{x}_i, \mathbf{w}^* \rangle \geq 1$. Note that, using the notation in Theorem 9.1, we therefore get

$$B = \min\{\|\mathbf{w}\| : \forall i \in [m],\;\; y_i \langle \mathbf{w}, x_i \rangle \geq 1\} \leq \sqrt{m}.$$

Thus, $(BR)^2 \leq m$.

- When running the Perceptron on this sequence of examples it makes m updates before converging.

Hint: Choose $d = m$ and for every i choose $\mathbf{x}_i = \mathbf{e}_i$.

9.4 **(*)** Given any number m, find an example of a sequence of labeled examples $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in (\mathbb{R}^3 \times \{-1, +1\})^m$ on which the upper bound of Theorem 9.1 equals m and the perceptron algorithm is bound to make m mistakes.
Hint: Set each $\mathbf{x}_i$ to be a third dimensional vector of the form (a, b, y_i), where $a^2 + b^2 = R^2 - 1$. Let $\mathbf{w}^*$ be the vector $(0, 0, 1)$. Now, go over the proof of the Perceptron's upper bound (Theorem 9.1), see where we used inequalities ($\leq$) rather than equalities ($=$), and figure out scenarios where the inequality actually holds with equality.

9.5 Suppose we modify the Perceptron algorithm as follows: In the update step, instead of performing $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + y_i \mathbf{x}_i$ whenever we make a mistake, we perform $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta y_i \mathbf{x}_i$ for some $\eta > 0$. Prove that the modified Perceptron will perform the same number of iterations as the vanilla Perceptron and will converge to a vector that points to the same direction as the output of the vanilla Perceptron.

9.6 In this problem, we will get bounds on the VC-dimension of the class of (closed) balls in $\mathbb{R}^d$, that is,

$$\mathcal{B}_d = \{B_{\mathbf{v},r} : \mathbf{v} \in \mathbb{R}^d, r > 0\},$$

where

$$B_{\mathbf{v},r}(\mathbf{x}) = \begin{cases} 1 & \text{if } \|\mathbf{x} - \mathbf{v}\| \leq r \\ 0 & \text{otherwise} \end{cases}.$$

1. Consider the mapping $\phi : \mathbb{R}^d \to \mathbb{R}^{d+1}$ defined by $\phi(\mathbf{x}) = (\mathbf{x}, \|\mathbf{x}\|^2)$. Show that if $\mathbf{x}_1, \ldots, \mathbf{x}_m$ are shattered by $\mathcal{B}_d$ then $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_m)$ are shattered by the class of halfspaces in $\mathbb{R}^{d+1}$ (in this question we assume that $\text{sign}(0) = 1$). What does this tell us about $\text{VCdim}(\mathcal{B}_d)$?
2. **(*)** Find a set of $d + 1$ points in $\mathbb{R}^d$ that is shattered by $\mathcal{B}_d$. Conclude that

$$d + 1 \leq \text{VCdim}(\mathcal{B}_d) \leq d + 2.$$