Chapter 7
Regularization

Keywords Data Augmentation · Robustness · Semi-Supervised Learning · Transfer Learning · Multitask Learning

Many ML methods use the principle of ERM (see Chap. 4) to learn a hypothesis out of a hypothesis space by minimizing the average loss (training error) on a set of labeled data points (training set). Using ERM as a guiding principle for ML methods makes sense only if the training error is a good indicator for its loss incurred outside the training set.

Figure 7.1 illustrates a typical scenario for a modern ML method which uses a large hypothesis space. This large hypothesis space includes highly non-linear maps which can perfectly resemble any dataset of modest size. However, there might be non-linear maps for which a small training error does not guarantee accurate predictions for the labels of data points outside the training set.

Chapter 6 discussed validation techniques to verify if a hypothesis with small training error will predict also well the labels of data points outside the training set. These validation techniques, including Algorithms 5 and 6, probe the hypothesis $\hat{h} \in \mathcal{H}$ delivered by ERM on a validation set. The validation set consists of data points which have not been used for the training set of ERM (4.3). The validation error, which is the average loss of the hypothesis on the data points in the validation set, serves as an estimate for the average error or risk (4.1) of the hypothesis $\hat{h}$.

This chapter discusses regularization as an alternative to validation techniques. In contrast to validation, regularization techniques do not require having a separate validation set which is not used for the ERM (4.3). This makes regularization attractive for applications where obtaining a separate validation set is difficult or costly (where labelled data is scarce).

Instead of probing a hypothesis $\hat{h}$ on a validation set, regularization techniques compute estimate the loss increase when applying $\hat{h}$ to data points outside the training set. The loss increase is estimated by adding a regularization term to the training error in ERM (4.3).

A. Jung, *Machine Learning*, Machine Learning: Foundations, Methodologies, and Applications, https://doi.org/10.1007/978-981-16-8193-6_7

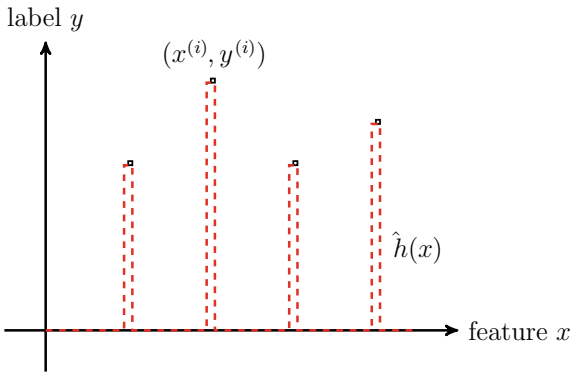

Fig. 7.1 The non-linear hypothesis map $\hat{h}$ perfectly fits the training set and has vanishing training error. Despite perfectly fitting the training set, the hypothesis $\hat{h}$ delivers the trivial (and useless) prediction $\hat{y} = \hat{h}(x) = 0$ for any datapoint that is not in the vicinity of the data points in the training set

Section 7.1 discusses the resulting regularized ERM, which we will refer to as structural risk minimization (SRM). It turns out that the SRM is equivalent to ERM using a smaller (pruned) hypothesis space. The amount of pruning depends on the weight of the regularization term relative to the training error. For an increasing weight of the regularization term, we obtain a stronger pruning resulting in a smaller effective hypothesis space.

Section 7.2 constructs regularization terms by requiring the resulting ML method to be robust against (small) random perturbations of the data points in a training set. Here, we replace each data point of a training set by the realization of a RV that fluctuates around this data point. This construction allows to interpret regularization as a (implicit) form of data augmentation.

Section 7.3 discusses data augmentation methods as a simulation-based implementation of regularization. Data augmentation adds a certain number of perturbed copies to each data point in the training set. One way to construct perturbed copies of a data point is to add (the realization of) a random vector to its features.

Section 7.4 analyzes the effect of regularization for linear regression using a simple probabilistic model for data points. This analysis parallels our previous study of the validation error of linear regression in Sect. 6.4. Similar to Sect. 6.4, we reveal a trade-off between the bias and variance of the hypothesis learnt by regularized linear regression. This trade- off was traced out by a discrete model parameter (the effective dimension) in Sect. 6.4. In contrast, regularization offers a continuous trade-off between bias and variance via a continuous regularization parameter.

Semi-supervised learning (SSL) uses (large amounts of) unlabeled data points to support the learning of a hypothesis from (a small number of) labeled data points [1]. Section 7.5 discusses SSL methods that use the statistical properties of unlabeled data points to construct useful regularization terms. These regularization terms are then used in SRM with a (typically small) set of labeled data points.

Multitask learning exploits similarities between different but related learning tasks [2]. We can formally define a learning task by a particular choice for the loss function (loss function) (see Sect. 2.3). The primary role of a loss function is to score the quality of a hypothesis map. However, the loss function also encapsulates the choice

for the label of a data point. For learning tasks defined for a single underlying data generation process it is reasonable to assume that the same subset of features is relevant for those learning tasks. One example for such related learning tasks is a multi-label classification problem (see Section) where each individual label of a data point represents an separate learning task. Section 7.6 shows how multitask learning can be implemented using regularization methods. The loss incurred in different learning tasks serves mutual regularization terms in a joint SRM for all learning tasks.

Section 7.7 shows how regularization can be used for **transfer learning**. Like multitask learning also transfer learning exploits relations between different learning tasks. In contrast to multitask learning, which jointly solves the individual learning tasks, transfer learning solves the learning tasks sequentially. The most basic form of transfer learning is to fine tune a pre-trained model. A pre-trained model can be obtained via ERM (4.3) in a ("source") learning task for which we have a large amount of labeled training data. The fine-tuning is then obtained via ERM (4.3) in the ("target") learning task of interest for which we might have only a small amount of labeled training data.

7.1 Structural Risk Minimization

Section 2.2 defined the effective dimension $d_{\rm eff}(\mathcal{H})$ of a hypothesis space $\mathcal{H}$ as the maximum number of data points that can be perfectly fit by some hypothesis $h \in \mathcal{H}$. As soon as the effective dimension of the hypothesis space in (4.3) exceeds the number m of training data points, we can find a hypothesis that perfectly fits the training data. However, a hypothesis that perfectly fits the training data might deliver poor predictions for data points outside the training set (see Fig. 7.1).

Modern ML methods typically use a hypothesis space with large effective dimension [3, 4]. Two well-known examples for such methods is linear regression (see Sect. 3.1) using a large number of features and deep learning with ANNs using a large number (billions) of artificial neurons (see Section 3.11). The effective dimension of these methods can be easily on the order of billions (10^9) if not larger [5]. To avoid overfitting during the naive use of ERM (4.3) we would require a training set containing at least as many data points as the effective dimension of the hypothesis space. However, in practice we often do not have access to training sets containing billions of labeled data points.

It seems natural to combat overfitting of a ML method by pruning its hypothesis space $\mathcal{H}$. We prune $\mathcal{H}$ by removing some of the hypothesis in $\mathcal{H}$ to obtain the smaller hypothesis space $\mathcal{H}' \subset \mathcal{H}$. We then replace ERM (4.3) with the restricted (or pruned) ERM

$$\hat{h} = \operatorname*{argmin}_{h \in \mathcal{H}'} \widehat{L}(h|\mathcal{D}) \text{ with pruned hypothesis space } \mathcal{H}' \subset \mathcal{H}. \tag{7.1}$$

The effective dimension of the pruned hypothesis space $\mathcal{H}'$ is typically much smaller than the effective dimension of the original (large) hypothesis space $\mathcal{H}$, $d_{\rm eff}\left(\mathcal{H}'\right) \ll d_{\rm eff}\left(\mathcal{H}\right)$. For a given size m of the training set, the risk of overfitting in (7.1) is much smaller than the risk of overfitting in (4.3).

Example. Consider linear regression which the hypothesis space (3.1) constituted by linear maps $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. The effective dimension of (3.1) is equal to the number of features, $d_{\rm eff}\left(\mathcal{H}\right) = n$. The hypothesis space $\mathcal{H}$ might be too large if we use a large number n of features, leading to overfitting. We prune (3.1) by retaining only linear hypotheses $h(\mathbf{x}) = \left(\mathbf{w}'\right)^T\mathbf{x}$ with weight vectors $\mathbf{w}'$ satisfying $w'_3 = \mathbf{w}'_4 = \ldots = \mathbf{w}'_n = 0$. Thus, the hypothesis space $\mathcal{H}'$ is constituted by all linear maps that only depend on the first two features x_1, x_2 of a data point. The effective dimension of $\mathcal{H}'$ is dimension is $d_{\rm eff}\left(\mathcal{H}'\right) = 2$ instead of $d_{\rm eff}\left(\mathcal{H}\right) = n$.

Pruning the hypothesis space is a special case of a more general strategy which we refer to as SRM [6]. The idea behind SRM is to modify the training error in ERM (4.3) to favour hypotheses which are more smooth or regular in a specific sense. By enforcing a smooth hypothesis, a ML methods becomes less sensitive, or more robust, to small perturbations of the training data points. Section 7.2 discusses the intimate relation between the robustness (against perturbations of the training set) of a ML method and its ability to generalize to data points outside the training set.

We measure the smoothness of a hypothesis using a regularizer $\mathcal{R}(h) \in \mathbb{R}_+$. Roughly speaking, the value $\mathcal{R}(h)$ measures the irregularity or variation of a predictor map h. The (design) choice for the regularizer depends on the precise definition of what is meant by regularity or variation of a hypothesis. Section 7.3 discusses how a particular choice for the regularizer $\mathcal{R}(h)$ arises naturally from a probabilistic model for data points.

We obtain SRM by adding the scaled regularizer $\lambda\mathcal{R}(h)$ to the ERM (4.3),

$$\begin{aligned}\hat{h} &= \operatorname*{argmin}_{h\in\mathcal{H}} \left[\widehat{L}(h|\mathcal{D}) + \lambda\mathcal{R}(h)\right] \\ &\overset{(2.16)}{=} \operatorname*{argmin}_{h\in\mathcal{H}} \left[(1/m)\sum_{i=1}^{m} L((\mathbf{x}^{(i)}, y^{(i)}), h) + \lambda\mathcal{R}(h)\right]. \end{aligned} \tag{7.2}$$

We can interpret the penalty term $\lambda\mathcal{R}(h)$ in (7.2) as an estimate (or approximation) for the increase, relative to the training error on $\mathcal{D}$, of the average loss of a hypothesis $\hat{h}$ when it is applied to data points outside $\mathcal{D}$. Another interpretation of the term $\lambda\mathcal{R}(h)$ will be discussed in Sect. 7.3.

The regularization parameter λ allows us to trade between a small training error $\widehat{L}(h^{(\mathbf{w})}|\mathcal{D})$ and small regularization term $\mathcal{R}(h)$, which enforces smoothness or regularity of h. If we choose a large value for λ, irregular or hypotheses h, with large $\mathcal{R}(h)$, are heavily "punished" in (7.2). Thus, increasing the value of λ results in the solution (minimizer) of (7.2) having smaller $\mathcal{R}(h)$. On the other hand, choosing a small value for λ in (7.2) puts more emphasis on obtaining a hypothesis h incurring a small training error. For the extreme case $\lambda = 0$, the SRM (7.2) reduces to ERM (4.3).

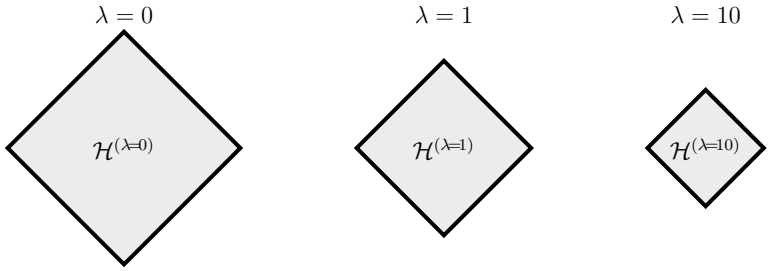

Fig. 7.2 Adding the scaled regularizer $\lambda\mathcal{R}(h)$ to the training error in the objective function of SRM (7.2) is equivalent to solving ERM (7.1) with a pruned hypothesis space $\mathcal{H}^{(\lambda)}$

The pruning approach (7.1) is intimately related to the SRM (7.2). They are, in a certain sense, **dual** to each other. First, note that (7.2) reduces to the pruning approach (7.1) when using the regularizer $\mathcal{R}(h) = 0$ for all $h \in \mathcal{H}'$, and $\mathcal{R}(h) = \infty$ otherwise, in (7.2). In the other direction, for many important choices for the regularizer $\mathcal{R}(h)$, there is a restriction $\mathcal{H}^{(\lambda)} \subset \mathcal{H}$ such that the solutions of (7.1) and (7.2) coincide (see Fig. 7.2). The relation between the optimization problems (7.1) and (7.2) can be made precise using the theory of convex duality (see [7, Ch. 5] and [8]).

For a hypothesis space $\mathcal{H}$ whose elements $h \in \mathcal{H}$ are parameterized by a weight vector $\mathbf{w} \in \mathbb{R}^n$, we can rewrite SRM (7.2) as

$$\begin{aligned}\widehat{\mathbf{w}}^{(\lambda)} &= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \big[\widehat{L}(h^{(\mathbf{w})}|\mathcal{D}) + \lambda\mathcal{R}(\mathbf{w})\big] \\ &= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \big[(1/m)\sum_{i=1}^{m} L((\mathbf{x}^{(i)}, y^{(i)}), h^{(\mathbf{w})}) + \lambda\mathcal{R}(\mathbf{w})\big]. \end{aligned} \tag{7.3}$$

For the particular choice of squared error loss (2.8), linear hypothesis space (3.1) and regularizer $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_2^2$, SRM (7.3) specializes to

$$\widehat{\mathbf{w}}^{(\lambda)} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \big[(1/m)\sum_{i=1}^{m} \big(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\big)^2 + \lambda\|\mathbf{w}\|_2^2\big]. \tag{7.4}$$

The special case (7.4) of SRM (7.3) is known as ridge regression [9].

Ridge regression (7.4) is equivalent to (see [8, Ch. 5])

$$\widehat{\mathbf{w}}^{(\lambda)} = \operatorname*{argmin}_{h^{(\mathbf{w})}\in\mathcal{H}^{(\lambda)}} (1/m)\sum_{i=1}^{m} \big(y^{(i)} - h^{(\mathbf{w})}(\mathbf{x}^{(i)})\big)^2 \tag{7.5}$$

with the restricted hypothesis space

$$\mathcal{H}^{(\lambda)} := \{h^{(\mathbf{w})} : \mathbb{R}^n \to \mathbb{R} : h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}, \\ \text{with some weights (weights) } \mathbf{w} \text{ satisfying } \|\mathbf{w}\|_2^2 \leq C(\lambda)\} \subset \mathcal{H}^{(n)}. \quad (7.6)$$

For any given value λ of the regularization parameter in (7.4), there is a number $C(\lambda)$ such that solutions of (7.4) coincide with the solutions of (7.5). Thus, ridge regression (7.4) i is equivalent to linear regression using a pruned version $\mathcal{H}^{(\lambda)}$ of the linear hypothesis space (3.1). The pruned hypothesis space $\mathcal{H}^{(\lambda)}$ (7.6) depends varies continuously with the regularization parameter λ.

Another popular special case of ERM (7.3) is obtained for the regularizer $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_1$ and known as the Lasso [10]

$$\widehat{\mathbf{w}}^{(\lambda)} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \big[(1/m)\sum_{i=1}^{m}\big(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\big)^2 + \lambda\|\mathbf{w}\|_1\big]. \quad (7.7)$$

Ridge regression (7.4) and the Lasso (7.7) have fundamentally different computational and statistical properties. Involving a smooth and convex objective function, ridge regression (7.4) can be implemented using efficient GD methods. The objective function of Lasso (7.7) is also convex but non-smooth and therefore requires advanced optimization methods. The increased computational complexity of Lasso (7.7) comes at the benefit of typically delivering a hypothesis with a smaller risk than those obtained from ridge regression [4, 10].

7.2 Robustness

Section 7.1 motivates regularization as a soft variant of model selection. Indeed, the regularization term in SRM (7.2) is equivalent to ERM (7.1) using a pruned (reducing) hypothesis space. We now discuss an alternative view on regularization as a means to make ML methods robust.

The ML methods discussed in Chap. 4 rest on the idealizing assumption that we have access to the true label values and feature values of labeled data points (the training set). These methods learn a hypothesis $h \in \mathcal{H}$ with minimum average loss (training error) incurred for data points in the training set. In practice, the acquisition of label and feature values might be prone to errors. These errors might stem from the measurement device itself (hardware failures or thermal noise) or might be due to human mistakes such as labelling errors.

Let us assume for the sake of exposition that the label values $y^{(i)}$ in the training set are accurate but that the features $\mathbf{x}^{(i)}$ are a perturbed version of the true features of the ith data point. Thus, instead of having observed the data point $\big(\mathbf{x}^{(i)}, y^{(i)}\big)$ we could have equally well observed the data point $\big(\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}, y^{(i)}\big)$ in the training set. Here, we have modelled the perturbations in the features using a RV $\boldsymbol{\varepsilon}$. The probability distribution of the perturbation $\boldsymbol{\varepsilon}$ is a design parameter that controls

robustness properties of the overall ML method. We will study a particular choice for this distribution in Sect. 7.3.

A robust ML method should learn a hypothesis that incurs a small loss not only for a specific data point $\big(\mathbf{x}^{(i)}, y^{(i)}\big)$ but also for perturbed data points $\big(\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}, y^{(i)}\big)$. Therefore, it seems natural to replace the loss $L\big(\big(\mathbf{x}^{(i)}, y^{(i)}\big), h\big)$, incurred on the ith data point in the training set, with the expectation

$$\mathbb{E}\big\{L\big(\big(\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}, y^{(i)}\big), h\big)\big\}. \tag{7.8}$$

The expectation (7.8) is computed using the probability distribution of the perturbation $\boldsymbol{\varepsilon}$. We will show in Sect. 7.3 that minimizing the average of the expectation (7.8), for $i = 1, \ldots, m$, is equivalent to the SRM (7.2).

Using the expected loss (7.8) is not the only possible approach to make a ML method robust. Another approach to make a ML method robust is known as bagging. The idea of bagging is to use the bootstrap method (see Sect. 6.5 and [9, Chap. 8]) to construct a finite number of perturbed copies $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(B)}$ of the original training set $\mathcal{D}$.

We then learn (e.g, using ERM) a separate hypothesis $h^{(b)}$ for each perturbed copy $\mathcal{D}^{(b)}$, $b = 1, \ldots, B$. This results in a whole ensemble of different hypotheses $h^{(b)}$ which might even belong to different hypothesis spaces. For example, one the hypotheis $h^{(1)}$ could be a linear map (see Sect. 3.1) and the hypothesis $h^{(2)}$ could be obtained from an ANN (see Sect. 3.11).

The final hypothesis delivered by bagging is obtained by combining or aggregating (e.g., using the average) the predictions $h^{(b)}\big(\mathbf{x}\big)$ delivered by each hypothesis $h^{(b)}$, for $b = 1, \ldots, B$ in the ensemble. The ML method referred to as random forest uses bagging to learn an ensemble of decision trees (see Sect. 3.10). The individual predictions obtained from the trees in a random forest are combined (e.g., using an average in regression or a majority vote in binary classification) to obtain a final prediction [9].

7.3 Data Augmentation

ML methods using ERM (4.3) are prone to overfitting as soon as the effective dimension of the hypothesis space $\mathcal{H}$ exceeds the number m of training data points. Sections 6.3 and 7.1 approached this by modifying either the model or the loss function by adding a regularization term. Both approaches prune the hypothesis space $\mathcal{H}$ underlying a ML method to reduce the effective dimension $d_{\text{eff}}\,(\mathcal{H})$. Model selection does this reduction in a discrete fashion while regularization implements a soft "shrinking" of the hypothesis space.

Instead of trying to reduce the effective dimension we could also try to increase the number m of training data points used in ERM (4.3). We now discuss how to synthetically generate new labeled data points by exploiting known structures that are inherent to a given application domain.

The data arising in many ML applications exhibit intrinsic symmetries and invariances at least in some approximation. The rotated image of a cat still shows a cat. The temperature measurement taken at a given location will be similar to another measurement taken 10 milliseconds later. Data augmentation exploits such symmetries and invariances to augment the raw data with additional synthetic data.

Let us illustrate data augmentation using an application that involves data points characterized by features $\mathbf{x} \in \mathbb{R}^n$ and number labels $y \in \mathbb{R}$. We assume that the data generating process is such that data points with close feature values have the same label. Equivalently, this assumption is requiring the resulting ML method to be robust against small perturbations of the feature values (see Sect. 7.2). This suggests to augment a data point $\big(\mathbf{x}, y\big)$ by several synthetic data points

$$\big(\mathbf{x} + \boldsymbol{\varepsilon}^{(1)}, y\big), \ldots, \big(\mathbf{x} + \boldsymbol{\varepsilon}^{(B)}, y\big), \tag{7.9}$$

with $\boldsymbol{\varepsilon}^{(1)}, \ldots, \boldsymbol{\varepsilon}^{(B)}$ being realizations of independent and identically distributed (i.i.d.) (i.i.d.) random vectors with the same probability distribution $p(\boldsymbol{\varepsilon})$.

Given a (raw) dataset $\mathcal{D} = \big\{\big(\mathbf{x}^{(1)}, y^{(1)}\big), \ldots, \big(\mathbf{x}^{(m)}, y^{(m)}\big)\big\}$ we denote the associated augmented dataset by

$$\begin{aligned} \mathcal{D}' = \big\{ & \big(\mathbf{x}^{(1,1)}, y^{(1)}\big), \ldots, \big(\mathbf{x}^{(1,B)}, y^{(1)}\big), \\ & \big(\mathbf{x}^{(2,1)}, y^{(2)}\big), \ldots, \big(\mathbf{x}^{(2,B)}, y^{(2)}\big), \\ & \ldots \\ & \big(\mathbf{x}^{(m,1)}, y^{(m)}\big), \ldots, \big(\mathbf{x}^{(m,B)}, y^{(m)}\big)\big\}. \end{aligned} \tag{7.10}$$

The size of the augmented dataset $\mathcal{D}'$ is $m' = B \times m$. For a sufficiently large augmentation parameter B, the augmented sample size m' is larger than the effective dimension n of the hypothesis space $\mathcal{H}$. We then learn a hypothesis via ERM on the augmented dataset,

$$\begin{aligned} \hat{h} &= \operatorname*{argmin}_{h \in \mathcal{H}} \widehat{L}(h|\mathcal{D}') \\ &\overset{(7.10)}{=} \operatorname*{argmin}_{h \in \mathcal{H}} (1/m') \sum_{i=1}^{m} \sum_{b=1}^{B} L((\mathbf{x}^{(i,b)}, y^{(i,b)}), h) \\ &\overset{(7.9)}{=} \operatorname*{argmin}_{h \in \mathcal{H}} (1/m) \sum_{i=1}^{m} (1/B) \sum_{b=1}^{B} L((\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}^{(b)}, y^{(i)}), h). \end{aligned} \tag{7.11}$$

We can interpret data-augmented ERM (7.11) as a data-driven form of regularization (see Sect. 7.1). The regularization is implemented by replacing, for each data point $\big(\mathbf{x}^{(i)}, y^{(i)}\big) \in \mathcal{D}$, the loss $L((\mathbf{x}^{(i)}, y^{(i)}), h)$ with the average loss $(1/B) \sum_{b=1}^{B} L((\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}^{(b)}, y^{(i)}), h)$ over the augmented data points that accompany $\big(\mathbf{x}^{(i)}, y^{(i)}\big) \in \mathcal{D}$.

Note that in order to implement (7.11) we need to first generate B realizations $\boldsymbol{\varepsilon}^{(b)} \in \mathbb{R}^n$ of i.i.d. random vectors with common probability distribution $p(\boldsymbol{\varepsilon})$. This might be computationally costly for a large B, n. However, when using a large augmentation parameter B, we might use the approximation

$$(1/B)\sum_{b=1}^{B} L((\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}^{(b)}, y^{(i)}), h) \approx \mathbb{E}\big\{L((\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}, y^{(i)}), h)\big\}. \tag{7.12}$$

This approximation is made precise by a key result of probability theory, known as the law of large numbers. We obtain an instance of ERM by inserting (7.12) into (7.11),

$$\hat{h} = \operatorname*{argmin}_{h\in\mathcal{H}} (1/m)\sum_{i=1}^{m} \mathbb{E}\big\{L((\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}, y^{(i)}), h)\big\}. \tag{7.13}$$

The usefulness of (7.13) as an approximation to the augmented ERM (7.11) depends on the difficulty of computing the expectation $\mathbb{E}\big\{L((\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}, y^{(i)}), h)\big\}$. The complexity of computing this expectation depends on the choice of loss function and the choice for the probability distribution $p(\boldsymbol{\varepsilon})$.

Let us study (7.13) for the special case linear regression with squared error loss (2.8) and linear hypothesis space (3.1),

$$\hat{h} = \operatorname*{argmin}_{h^{(\mathbf{w})}\in\mathcal{H}^{(n)}} (1/m)\sum_{i=1}^{m} \mathbb{E}\big\{\big(y^{(i)} - \mathbf{w}^T\big(\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}\big)\big)^2\big\}. \tag{7.14}$$

We use perturbations $\boldsymbol{\varepsilon}$ drawn a multivariate normal distribution with zero mean and covariance matrix $\sigma^2\mathbf{I}$,

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}). \tag{7.15}$$

We develop (7.14) further by using

$$\mathbb{E}\{\big(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\big)\boldsymbol{\varepsilon}\} = \mathbf{0}. \tag{7.16}$$

The identity (7.16) uses that the data points $\big(\mathbf{x}^{(i)}, y^{(i)}\big)$ are fixed and known (deterministic) while $\boldsymbol{\varepsilon}$ is a zero-mean random vector. Combining (7.16) with (7.14),

$$\begin{aligned}\mathbb{E}\big\{\big(y^{(i)} - \mathbf{w}^T\big(\mathbf{x}^{(i)} + \boldsymbol{\varepsilon}\big)\big)^2\big\} &= \big(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\big)^2 + \|\mathbf{w}\|_2^2\,\mathbb{E}\big\{\|\boldsymbol{\varepsilon}\|_2^2\big\} \\ &= \big(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\big)^2 + n\|\mathbf{w}\|^2\sigma^2. \end{aligned} \tag{7.17}$$

where the last step used $\mathbb{E}\big\{\|\boldsymbol{\varepsilon}\|_2^2\big\} \overset{(7.15)}{=} n\sigma^2$. Inserting (7.17) into (7.14),

$$\hat{h} = \operatorname*{argmin}_{h(\mathbf{w}) \in \mathcal{H}^{(n)}} (1/m) \sum_{i=1}^{m} \left(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}\right)^2 + n \left\|\mathbf{w}\right\|^2 \sigma^2. \tag{7.18}$$

We have obtained (7.18) as an approximation of the augmented ERM (7.11) for the special case of squared error loss (2.8) and the linear hypothesis space (3.1). This approximation uses the law of large numbers (7.12) and becomes more accurate for increasing augmentation parameter B.

Note that (7.18) is nothing but ridge regression (7.4) using the regularization parameter $\lambda = n\sigma^2$. Thus, we can interpret ridge regression as implicit data augmentation (7.10) by applying random perturbations (7.9) to the feature vectors in the original training set $\mathcal{D}$.

The regularizer $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_2^2$ in (7.18) arose naturally from the specific choice for the probability distribution (7.15) of the random perturbation $\boldsymbol{\varepsilon}^{(i)}$ in (7.9) and using the squared error loss. Other choices for this probability distribution or the loss function result in different regularizers.

Augmenting data points with random perturbations distributed according (7.15) treat the features of a data point independently. For application domains that generate data points with highly correlated features it might be useful to augment data points using random perturbations $\boldsymbol{\varepsilon}$ (see (7.9)) distributed as

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{C}). \tag{7.19}$$

The covariance matrix $\mathbf{C}$ of the perturbation $\boldsymbol{\varepsilon}$ can be chosen using domain expertise or estimated (see Sect. 7.5). Inserting the distribution (7.19) into (7.13),

$$\hat{h} = \operatorname*{argmin}_{h(\mathbf{w}) \in \mathcal{H}^{(n)}} \left[(1/m) \sum_{i=1}^{m} \left(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}\right)^2 + \mathbf{w}^T \mathbf{C} \mathbf{w} \right]. \tag{7.20}$$

Note that (7.20) reduces to ordinary ridge regression (7.18) for the choice $\mathbf{C} = \sigma^2 \mathbf{I}$.

7.4 Statistical and Computational Aspects of Regularization

The goal of this section is to develop a better understanding for the effect of the regularization term in SRM (7.3). We will analyze the solutions of ridge regression (7.4) which is the special case of SRM using the linear hypothesis space (3.1) and squared error loss (2.8). Using the feature matrix $\mathbf{X} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\right)^T$ and label vector $\mathbf{y} = (y^{(1)}, \ldots, y^{(m)})^T$, we can rewrite (7.4) more compactly as

$$\widehat{\mathbf{w}}^{(\lambda)} = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \left[(1/m) \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \right]. \tag{7.21}$$

The solution of (7.21) is given by

$$\widehat{\mathbf{w}}^{(\lambda)} = (1/m)((1/m)\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}. \tag{7.22}$$

For $\lambda = 0$, (7.22) reduces to the formula (6.17) for the optimal weights in linear regression (see (7.4) and (4.5)). Note that for $\lambda > 0$, the formula (7.22) is always valid, even when $\mathbf{X}^T\mathbf{X}$ is singular (not invertible). For $\lambda > 0$ the optimization problem (7.21) (and (7.4)) has the unique solution (7.22).

To study the statistical properties of the predictor $h^{(\widehat{\mathbf{w}}^{(\lambda)})}(\mathbf{x}) = \left(\widehat{\mathbf{w}}^{(\lambda)}\right)^T\mathbf{x}$ (see (7.22)) we use the probabilistic toy model (6.13), (6.15) and (6.16) that we used already in Sect. 6.4. We interpret the training data $\mathcal{D}^{(\text{train})} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ as realizations of i.i.d. RVs whose distribution is defined by (6.13), (6.15) and (6.16).

We can then define the average prediction error of ridge regression as

$$E_{\text{pred}}^{(\lambda)} := \mathbb{E}\left\{\left(y - h^{(\widehat{\mathbf{w}}^{(\lambda)})}(\mathbf{x})\right)^2\right\}. \tag{7.23}$$

As shown in Sect. 6.4, the error $E_{\text{pred}}^{(\lambda)}$ is the sum of three components: the bias, the variance and the noise variance σ^2 (see (6.27)). The bias of $\widehat{\mathbf{w}}^{(\lambda)}$ is

$$B^2 = \left\|(\mathbf{I} - \mathbb{E}\{(\mathbf{X}^T\mathbf{X} + m\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}\})\overline{\mathbf{w}}\right\|_2^2. \tag{7.24}$$

For sufficiently large size m of the training set, we can use the approximation

$$\mathbf{X}^T\mathbf{X} \approx m\mathbf{I} \tag{7.25}$$

such that (7.24) can be approximated as

$$\begin{aligned} B^2 &\approx \left\|(\mathbf{I} - (\mathbf{I} + \lambda\mathbf{I})^{-1})\overline{\mathbf{w}}\right\|_2^2 \\ &= \sum_{l=1}^{n} \frac{\lambda}{1+\lambda}\overline{w}_l^2. \end{aligned} \tag{7.26}$$

Let us compare the (approximate) bias term (7.26) of regularized linear regression with the bias term (6.23) of ordinary linear regression (which is the extreme case of ridge regression with $\lambda = 0$). The bias term (7.26) increases with increasing regularization parameter λ in ridge regression (7.4). In many relevant settings, the increase in bias is outweighed by the reduction in variance. The variance typically decreases with increasing λ as shown next.

The variance of ridge regression (7.4) satisfies

$$\begin{aligned} V = (\sigma^2/m^2)\times \\ \text{tr}\left\{\mathbb{E}\{((1/m)\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}((1/m)\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\}\right\}. \end{aligned} \tag{7.27}$$

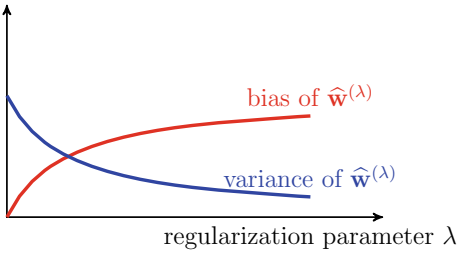

Fig. 7.3 The bias and variance of regularized linear regression depend on the regularization parameter λ in an opposite manner resulting in a bias-variance trade-off

Inserting the approximation (7.25) into (7.27),

$$V \approx \sigma^2(1/m)(n/(1+\lambda)). \tag{7.28}$$

According to (7.28), the variance of $\widehat{\mathbf{w}}^{(\lambda)}$ decreases with increasing regularization parameter λ of ridge regression (7.4). This is the opposite behaviour as observed for the bias (7.26), which increases with increasing λ. The approximate variance formula (7.28) suggests to interpret the ratio $(n/(1+\lambda))$ as the effective number of features used by ridge regression. Increasing the regularization parameter λ decreases the effective number of features.

Figure 7.3 illustrates the trade-off between the bias B^2 (7.26) of ridge regression, which increases for increasing λ, and the variance V (7.28) which decreases with increasing λ. Note that we have seen another example for a bias-variance trade-off in Sect. 6.4. This trade-off was traced out by a discrete (model complexity) parameter $r \in \{1, 2, \ldots\}$ (see (6.14)). In stark contrast to discrete model selection, the bias-variance trade-off for ridge regression is traced out by the continuous regularization parameter $\lambda \in \mathbb{R}_+$.

The main statistical effect of the regularization term in ridge regression is to balance the bias with the variance to minimize the average prediction error of the learnt hypothesis. There is also a computational effect or adding a regularization term. Roughly speaking, the regularization term serves as a pre-conditioning of the optimization problem and, in turn, reduces the computational complexity of solving ridge regression (7.21).

The objective function in (7.21) is a smooth (infinitely often differentiable) convex function. We can therefore use GD to solve (7.21) efficiently (see Chap. 5). Algorithm 8 summarizes the application of GD to (7.21). The computational complexity of Algorithm 8 depends crucially on the number of GD iterations required to reach a sufficiently small neighbourhood of the solutions to (7.21). Adding the regularization term $\lambda\|\mathbf{w}\|_2^2$ to the objective function of linear regression **speeds up GD**. To verify this claim, we first rewrite (7.21) as the quadratic problem

$$\min_{\mathbf{w}\in\mathbb{R}^n} \underbrace{(1/2)\mathbf{w}^T\mathbf{Q}\mathbf{w} - \mathbf{q}^T\mathbf{w}}_{=f(\mathbf{w})}$$
$$\text{with } \mathbf{Q} = (1/m)\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}, \mathbf{q} = (1/m)\mathbf{X}^T\mathbf{y}. \tag{7.29}$$

This is similar to the quadratic optimization problem (4.9) underlying linear regression but with a different matrix $\mathbf{Q}$. The computational complexity (number of iterations) required by GD (see (5.4)) applied to solve (7.29) up to a prescribed accuracy depends crucially on the condition number $\kappa(\mathbf{Q}) \geq 1$ of the psd matrix $\mathbf{Q}$ [11]. The smaller the condition number $\kappa(\mathbf{Q})$, the fewer iterations are required by GD. A matrix with small condition number is also referred to as being "well-conditioned".

The condition number of the matrix $\mathbf{Q}$ in (7.29) is given by

$$\kappa(\mathbf{Q}) = \frac{\lambda_{\max}((1/m)\mathbf{X}^T\mathbf{X}) + \lambda}{\lambda_{\min}((1/m)\mathbf{X}^T\mathbf{X}) + \lambda}. \tag{7.30}$$

According to (7.30), the condition number tends to one for increasing regularization parameter λ,

$$\lim_{\lambda\to\infty} \frac{\lambda_{\max}((1/m)\mathbf{X}^T\mathbf{X}) + \lambda}{\lambda_{\min}((1/m)\mathbf{X}^T\mathbf{X}) + \lambda} = 1. \tag{7.31}$$

Thus, the number of required GD iterations in Algorithm 8 decreases with increasing regularization parameter λ.

Algorithm 8 Regularized Linear regression via GD

Input: dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$; GD step size $\alpha > 0$.
Initialize:set $\mathbf{w}^{(0)} := \mathbf{0}$; set iteration counter $k := 0$
1: **repeat**
2: $r := r + 1$ (increase iteration counter)
3: $\mathbf{w}^{(r)} := (1 - \alpha\lambda)\mathbf{w}^{(r-1)} + \alpha(2/m)\sum_{i=1}^m (y^{(i)} - (\mathbf{w}^{(r-1)})^T\mathbf{x}^{(i)})\mathbf{x}^{(i)}$ (do a GD step (5.4))
4: **until** stopping criterion met
Output: $\mathbf{w}^{(r)}$ (which approximates $\widehat{\mathbf{w}}^{(\lambda)}$ in (7.21))

7.5 Semi-Supervised Learning

Consider the task of predicting the numeric label y of a data point $\mathbf{z} = (\mathbf{x}, y)$ based on its feature vector $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$. At our disposal are two datasets $\mathcal{D}^{(u)}$ and $\mathcal{D}^{(l)}$. For each data point in $\mathcal{D}^{(u)}$ we only know the feature vector. We therefore refer to $\mathcal{D}^{(u)}$ as "unlabelled data". For each data point in $\mathcal{D}^{(l)}$ we know both, the feature vector $\mathbf{x}$ and the label y. We therefore refer to $\mathcal{D}^{(l)}$ as "labeled data".

SSL methods exploit the information provided by unlabelled data $\mathcal{D}^{(u)}$ to support the learning of a hypothesis based on minimizing its empirical risk on the labelled (training) data $\mathcal{D}^{(l)}$. The success of SSL methods depends on the statistical properties of the data generated within a given application domain. Loosely speaking, the information provided by the probability distribution of the features must be relevant for the ultimate task of predicting the label y from the features $\mathbf{x}$ [1].

Let us design a SSL method, summarized in Algorithm 9 below, using the data augmentation perspective from Sect. 7.3. The idea is the augment the (small) labeled dataset $\mathcal{D}^{(l)}$ by adding random perturbations for the features vectors of data point in $\mathcal{D}^{(l)}$. This is reasonable for applications where feature vectors are subject to inherent measurement or modelling errors. Given a data point with vector $\mathbf{x}$ we could have equally well observed a feature vector $\mathbf{x} + \boldsymbol{\varepsilon}$ with some small random perturbation $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$. To estimate the covariance matrix $\mathbf{C}$, we use the sample covariance matrix of the feature vectors in the (large) unlabelled dataset $\mathcal{D}^{(u)}$. We then learn a hypothesis using the augmented (regularized) ERM (7.20).

Algorithm 9 A Semi-Supervised Learning Algorithm

Input: labeled dataset $\mathcal{D}^{(l)} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$; unlabeled dataset $\mathcal{D}^{(u)} = \{\widetilde{\mathbf{x}}^{(i)}\}_{i=1}^{m'}$

1: compute $\mathbf{C}$ via sample covariance on $\mathcal{D}^{(u)}$,

$$\mathbf{C} := (1/m') \sum_{i=1}^{m'} \big(\widetilde{\mathbf{x}}^{(i)} - \widehat{\mathbf{x}}\big)\big(\widetilde{\mathbf{x}}^{(i)} - \widehat{\mathbf{x}}\big)^T \text{ with } \widehat{\mathbf{x}} := (1/m') \sum_{i=1}^{m'} \widetilde{\mathbf{x}}^{(i)}. \tag{7.32}$$

2: compute (e.g. using GD steps (5.4))

$$\widehat{\mathbf{w}} := \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \Big[(1/m) \sum_{i=1}^{m} \big(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}\big)^2 + \mathbf{w}^T \mathbf{C} \mathbf{w} \Big]. \tag{7.33}$$

Output: hypothesis $\hat{h}(\mathbf{x}) = \big(\widehat{\mathbf{w}}\big)^T \mathbf{x}$

7.6 Multitask Learning

We can identify a learning task with the loss function $L((\mathbf{x}, y), h)$ that is used to measure the quality of a particular hypothesis $h \in \mathcal{H}$. Note that the loss obtained for a given data point also depends on the definition for the label of a data point. For the same data points, we obtain different learning tasks from different choices or definitions for the label of a data point. Multitask learning exploits the similarities between different learning tasks to jointly solve them.

Example. Consider a data point $\mathbf{z}$ representing a hand-drawing that is collected via the online game https://quickdraw.withgoogle.com/. The features of a data point are the pixel intensities of the bitmap which is used to store the hand-drawing. As

label we could use the fact if a hand-drawing shows an apple or not. This results in the learning task $\mathcal{T}^{(1)}$. Another choice for the label of a hand-drawing could be the fact if a hand-drawing shows a fruit at all or not. This results in another learning task $\mathcal{T}^{(2)}$ which is similar but different from the task $\mathcal{T}^{(1)}$.

The idea of multitask learning is that a reasonable hypothesis h for a learning task should also do well for a related learning tasks. Thus, we can use the loss incurred on similar learning tasks as a regularization term for learning a hypothesis for the learning task at hand. Algorithm 10 is a straightforward implementation of this idea for a given dataset that gives rise to T related learning tasks $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(T)}$. For each individual learning task $\mathcal{T}^{(t')}$ it uses the loss on the remaining learning tasks $\mathcal{T}^{(t)}$, with $t \neq t'$, as regularization term in (7.34).

Algorithm 10 A Multitask Learning Algorithm

Input: dataset $\mathcal{D} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ with T associated learning tasks with loss functions $L^{(1)}, \ldots, L^{(T)}$, hypothesis space $\mathcal{H}$

1: learn a hypothesis $\hat{h}$ via

$$\hat{h} := \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{t=1}^{T} \sum_{i=1}^{m} L^{(t)}\big(\mathbf{z}^{(i)}, h\big). \tag{7.34}$$

Output: hypothesis $\hat{h}$

The applicability of Algorithm 10 is somewhat limited as it aims at finding a single hypothesis that does well for all T learning tasks simultaneously. For certain application domains it might be more reasonable to not learn a single hypothesis for all learning tasks but to learn a separate hypothesis $h^{(t)}$ for each learning task $t = 1, \ldots, T$. However, these separate hypotheses typically might still share some structural similarities.[1] We can enforce different notion of similarities between the hypotheses $h^{(t)}$ by adding a regularization term to the loss functions of the tasks.

Algorithm 11 generalizes Algorithms 10 by learning a separate hypothesis for each task t while requiring these hypotheses to be structurally similar. The structural (dis-)similarity between the hypotheses is measured by a regularization term $\mathcal{R}$.

7.7 Transfer Learning

Regularization is also instrumental for transfer learning to capitalize on synergies between different related learning tasks [13, 14]. Transfer learning is enabled by constructing regularization terms for a learning task by using the result of a previous

[1] One important example for such a structural similarity in the case of linear predictors $h^{(t)}(\mathbf{x}) = \big(\mathbf{w}^{(t)}\big)^T \mathbf{x}$ is when the weight vectors $\mathbf{w}^{(T)}$ have a small joint support $\bigcup_{t=1,\ldots,T} \operatorname{supp}(w^{(t)})$. Requiring the weight vectors to have a small joint support is equivalent to requiring the stacked vector $\widetilde{\mathbf{w}} = \big(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(T)}\big)$ to be block (group) sparse [12].

Algorithm 11 A Multitask Learning Algorithm

Input: dataset $\mathcal{D} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ with T associated learning tasks with loss functions $L^{(1)}, \ldots, L^{(T)}$, hypothesis space $\mathcal{H}$

1: learn a hypothesis $\hat{h}$ via

$$\hat{h}^{(1)}, \ldots, \hat{h}^{(T)} := \operatorname*{argmin}_{h^{(1)},\ldots,h^{(T)} \in \mathcal{H}} \sum_{t=1}^{T} \sum_{i=1}^{m} L^{(t)}\big(\mathbf{z}^{(i)}, h^{(t)}\big) + \lambda \mathcal{R}\big(h^{(1)}, \ldots, h^{(T)}\big). \tag{7.35}$$

Output: hypotheses $\hat{h}^{(1)}, \ldots, \hat{h}^{(T)}$

leaning task. While multitask learning methods solve many related learning tasks simultaneously, transfer learning methods operate in a sequential fashion.

To illustrate the idea of transfer learning consider two learning tasks which differ in their intrinsic difficulty. We consider a learning task to be easy if it involves if we can easily gather large amounts of labeled (training) data for that task. Consider the learning task $\mathcal{T}^{(1)}$ of predicting whether an image shows a cat or not. For this learning task we can easily gather a large training set $\mathcal{D}^{(1)}$ using via image collections of animals. Another (related) learning task $\mathcal{T}^{(2)}$ is to predicting whether an image shows a cat of a particular breed, with a particular body height and with a specific age, we might not be able to collect many labeled data points.

7.8 Exercises

Exercise 7.1 Ridge Regression is a Quadratic Problem. Consider the linear hypothesis space consisting of linear maps parameterized by weights $\mathbf{w}$. We try to find the best linear map by minimizing the regularized average squared error loss (empirical risk) incurred on some labeled data points $(\mathbf{x}^{(1)}, y^{(1)})$, $(\mathbf{x}^{(2)}, y^{(2)})$, …, $(\mathbf{x}^{(m)}, y^{(m)})$. As the regularizer we use $\|\mathbf{w}\|^2$, yielding the following learning problem

$$\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}) = (1/m) \sum_{i=1}^{m} \big(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}\big) + \|\mathbf{w}\|_2^2.$$

Is it possible to rewrite the objective function $f(\mathbf{w})$ as a convex quadratic function $f(\mathbf{w}) = \mathbf{w}^T \mathbf{C}\mathbf{w} + \mathbf{b}\mathbf{w} + c$? If this is possible, how are the matrix $\mathbf{C}$, vector $\mathbf{b}$ and constant c related to the feature vectors and labels of the training data?

Exercise 7.2 Regularization or Model Selection. Consider data points, each characterized by $n = 10$ features $\mathbf{x} \in \mathbb{R}^n$ and a single numeric label y. We want to learn a linear hypothesis $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ by minimizing the average squared error on the training set $\mathcal{D}$ of size $m = 4$. We could learn such a hypothesis by two approaches. The first approach is to split the dataset into a training set and a validation set. Then we consider all models that consists of linear hypotheses with weight vectors having at

most two non-zero weights. Each of these models corresponds to a different subset of two weights that might be non-zero. Find the model resulting in the smallest validation errors (see Algorithm 5). Compute the average loss of the resulting optimal linear hypothesis on some data points that have neither been used for training nor for validation. Compare this average loss ("test error") with the average loss obtained on the same data points by the hypothesis learnt by ridge regression (7.4).

References

1. O. Chapelle, B. Schölkopf, A. Zien (eds.), *Semi-Supervised Learning* (The MIT Press, Cambridge, MA, 2006)
2. R. Caruana, Multitask learning. Mach. Learn. **28**(1), 41–75 (1997)
3. M. Wainwright, *High-Dimensional Statistics: A Non-Asymptotic Viewpoint* (Cambridge University Press, Cambridge, 2019)
4. P. Bühlmann, S. van de Geer, *Statistics for High-Dimensional Data* (Springer, New York, 2011)
5. S. Shalev-Shwartz, S. Ben-David, *Understanding Machine Learning—From Theory to Algorithms* (Cambridge University Press, Cambridge, 2014)
6. V.N. Vapnik, *The Nature of Statistical Learning Theory* (Springer, Berlin, 1999)
7. S. Boyd, L. Vandenberghe, *Convex Optimization* (Cambridge University Press, Cambridge, UK, 2004)
8. D.P. Bertsekas, *Nonlinear Programming*, 2nd edn. (Athena Scientific, Belmont, MA, 1999)
9. T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning* Springer Series in Statistics. (Springer, New York, 2001)
10. T. Hastie, R. Tibshirani, M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Its Generalizations* (CRC Press, Boca Raton, FL, 2015)
11. A. Jung, A fixed-point of view on gradient methods for big data. Frontiers in Applied Mathematics and Statistics **3**, 18 (2017)
12. Y.C. Eldar, P. Kuppinger, H. Bölcskei, Block-sparse signals: Uncertainty relations and efficient recovery. IEEE Trans. Signal Processing **58**(6), 3042–3054 (2010). (June)
13. S. Pan, Q. Yang, A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010)
14. J. Howard, S. Ruder, Universal language model fine-tuning for text classification, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Association for Computational Linguistics, Stroudsburg, 2018), pp. 328–339