Exercises
Introduction to Machine Learning
Spring 2022

Institute for Machine learning
Dept. of Computer Science, ETH Zürich
Prof. Dr. Andreas Krause, Prof. Dr. Fanny Yang

Homework 2 (Classification, Overfitting)

For questions, please refer to Moodle.
Released on **22 March, 2022**

General Instructions

- Submission of solutions is not mandatory but solving the exercises are highly recommended. The master solution will be released next week.
- Part of the exercises are available on Moodle as a quiz. These problems are marked with [✓].

Exercise 1: True-False Classification with Asymmetric Losses

For a company's new spam email filter, they have designed two different classifiers to determine if an email is spam ($y = 1$) or non-spam ($y = -1$). Assessing the quality of a spam classifier is difficult. A standard $0 - 1$ classification loss does not account for the asymmetric business costs of incorrectly classifying a non-spam email as spam vs incorrectly classifying a spam email as non-spam. To address this, we set our null hypothesis to be that an email is non-spam and evaluate the two classifiers. Each classifier uses two features. Figure 1 illustrates the dataset with the two classifiers (A and B) each represented by a hard decision boundary that classifies the datapoints into two groups. The two classifiers classify the groups identically but using different decision boundaries. So first we will just study the performance of A.

(a) [✓] Compute the False Positive Rate (FPR) of classifier A.

Solution: See the lecture slides on classification for some of the formulas used here.

$$FPR = \frac{\#FP}{\#[y = -1]} = \frac{2}{6} = \frac{1}{3}$$

(b) [✓] Compute the False Discovery Rate (FDR) of classifier A.

Solution:

$$FDR = \frac{\#FP}{\#[\hat{y} = 1]} = \frac{2}{5}$$

(c) [✓] Compute the precision of classifier A.

Solution:

$$\text{Precision} = 1 - FDR = \frac{3}{5}$$

(d) [✓] Compute the recall of classifier A.

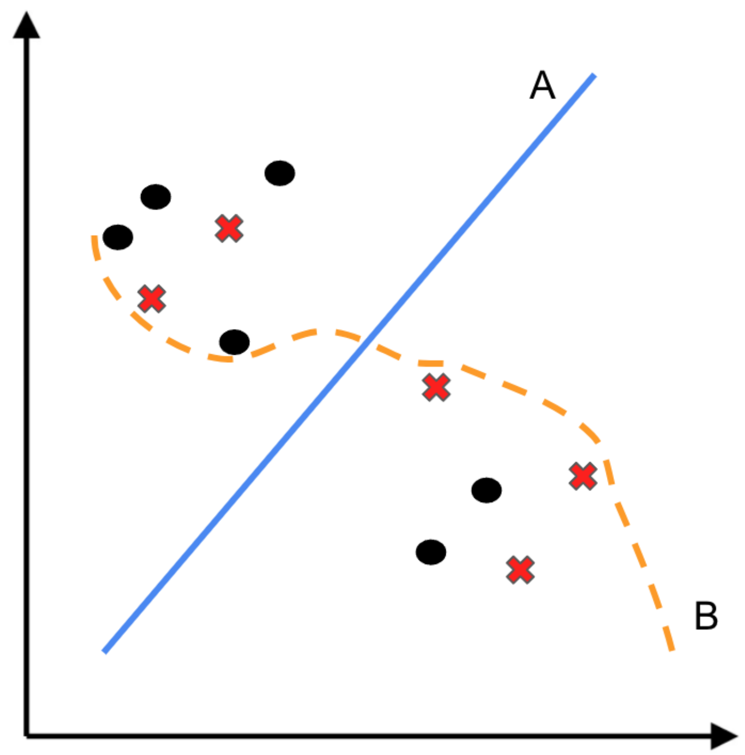

Figure 1: The '.' labels indicate that an email is non-spam ($y = -1$), 'x' indicates spam ($y = 1$). Everything to the right of line labelled A is classified by A as spam. Everything below the curve of B is classified as spam by B.

Solution:

$$\text{Recall} = \frac{\#TP}{\#[y = 1]} = \frac{3}{5}$$

(e) [✓] Compute the F1 Score of classifier A.

Solution: This is the harmonic mean of precision and recall so 3/5.

(f) [✓] Decide whether the following statements are True or False.

When comparing two classifiers on validation data, the best is always the classifier with highest F1 score — ☐ True [✓ **False**]

Solution: F1 is just one way to trade off between precision and recall. I could aggregate these two quantities in a different way and obtain a different highest scoring classifier. Moreover, F1 score does not take into account factors outside of precision and recall. For example fairness or adversarial robustness.

Classifier A could have been computed by hard-margin linear Support Vector Machine (SVM) without modifying the data, but not a soft margin SVM. — ☐ True [✓ **False**]

Solution: A is not a separating hyperplane (it miss-classifies some points) but with linear hard-margin SVM it is assumed that a linear separating hyperplane exists.

Classifier A is more robust than B to adversarial perturbations of the training datapoints. ■ **True** ☐ False

Solution: Changing the features by a small amount for any sets of datapoints is unlikely to effect the classifications given by A, because geometrically the corresponding line is far from any datapoints. Meanwhile for B one can see that slightly altering the features of some datapoints would negatively change the performance of the classifier.

The company decides that using a hard decision boundary is not the best way to trade-off costs in the spam filter problem. Instead they design 3 classifiers C, D, E that each use a threshold τ which controls the trade-off between false-positive rate (FPR) and true-discovery rate (TPR). For example, classifier C learns a prediction function $\hat{f}_C$ that maps from the input features x to $[0, 1]$. An email is then classified as spam if $\hat{f}_C(x) < \tau$. Figure 2 depicts the trade-off between FPR and True Positive Rate (TPR) for each classifier as a result of varying τ.

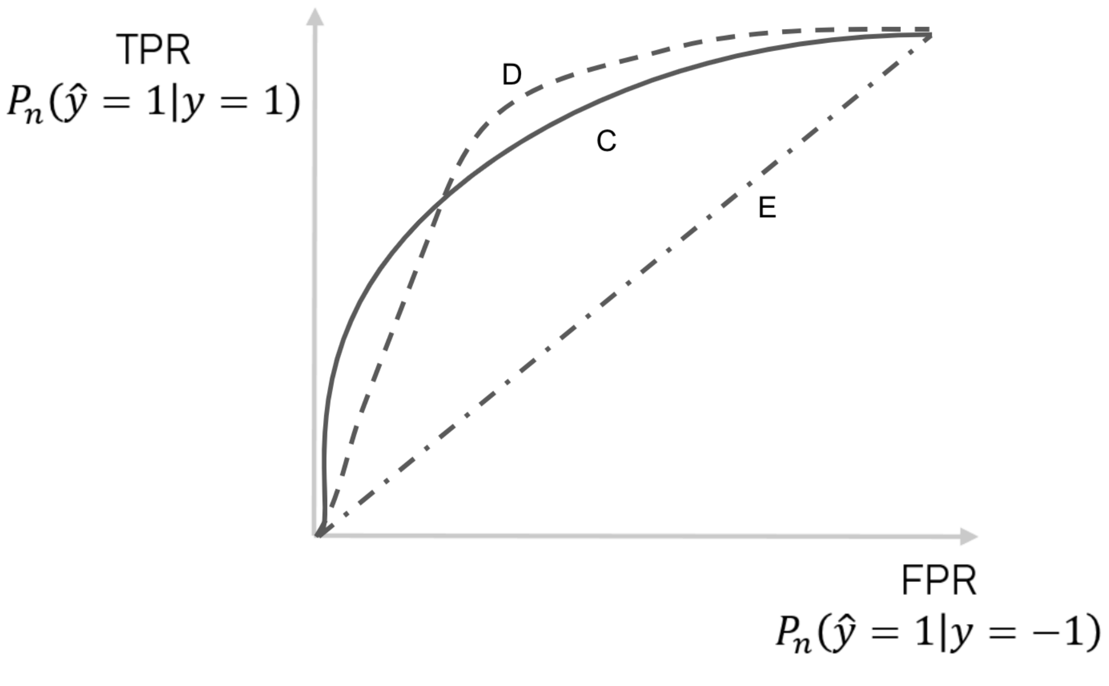

Figure 2: ROC curve for 3 different classifiers.

(g) [✓] For any of the classifiers, could $\hat{f}$ be independent of y?
1. No. 2. Yes: C. 3. Yes: D. 4. Yes: C and D. **5. Yes: E.**

Solution: Yes: E. f_C and f_D cannot be independent of y because $TPR > FPR$ implies that f contains some signal for predicting y. That is, a point classified positive is more likely to be correctly rather than incorrectly classified. Meanwhile E could be constructed by setting $f_E \sim \text{Unif}(0, 1)$: a point classified as positive for any τ is equally likely to be a true positive vs a false positive. If this is unintuitive, try designing a simple example based upon Figure 1.

(h) [✓] Out of the 3 classifiers, given that the desired tradeoff between FPR and TPR is unknown, what set contains all classifiers that *could* be the optimal classifier (only considering TPR and FPR as metrics)?

1. $\{C\}$ 2. $\{D\}$. 3. $\{E\}$ **4.** $\{C, D\}$ 5. $\{C, D, E\}$

Solution: $\{C, D\}$. For low desired FPRs, C has the highest TPR. But for high desired TPRs, D has the lowest FPRs. Meanwhile, E is dominated by both other classifiers for any value of τ since it has a lower TPR for any corresponding FPR.

Exercise 2: Ridge Regression

In the first homework, you have extensively worked with linear regression, i.e., the hypothesis space is affine functions. The goal of this exercise is to study the regularized version of this problem.

Let $D = \{(\boldsymbol{x}_1, y_1), \dots (\boldsymbol{x}_n, y_n)\}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ are the training data given to you. The ridge regression optimization problem with parameter $\lambda > 0$ is given by

$$\arg\min_{\boldsymbol{w}} L_{\text{ridge}}(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left[\sum_{i=1}^{n} \left(y_i - \boldsymbol{w}^\top \boldsymbol{x}_i \right)^2 + \lambda \boldsymbol{w}^\top \boldsymbol{w} \right]. \tag{1}$$

In this exercise, the $n \times d$ matrix $X \in \mathbb{R}^{n \times d}$ denotes a matrix with the $\boldsymbol{x}_i$ as its rows and the vector $\boldsymbol{y} \in \mathbb{R}^n$ consisting of the scalars y_i. Moreover, here, $\|\cdot\|$ is always the Euclidean norm. We refer to any $\boldsymbol{w}^*_{\text{ridge}}$ that attains the above minimum as a solution to the problem.

(a) Show that L_{ridge} has a positive semi-definite Hessian.

(b) A continuously differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is called *α-strongly convex* for some $\alpha > 0$, if for any points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ one has

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) + \frac{\alpha}{2} \|\boldsymbol{y} - \boldsymbol{x}\|^2.$$

If f is twice differentiable, an equivalent condition is that for any point $\boldsymbol{x} \in \mathbb{R}^d$, one has

$$D^2 f(\boldsymbol{x}) \succeq \alpha I,$$

which means $D^2 f(\boldsymbol{x}) - \alpha I$ is positive semi-definite for all $\boldsymbol{x} \in \mathbb{R}^d$.
Prove that a strongly convex function admits a unique minimizer in $\mathbb{R}^d$.
Hint: First prove that $f(\boldsymbol{x}) \to \infty$ as $\|\boldsymbol{x}\| \to \infty$ to show that there is some minimizer.

(c) Use (b) to show that (1) admits the unique solution $\boldsymbol{w}^*_{\text{ridge}}$ for any matrix X.

(d) What is the role of the term $\lambda \boldsymbol{w}^\top \boldsymbol{w}$ in L_{ridge}? What happens to $\boldsymbol{w}^*_{\text{ridge}}$ as $\lambda \to 0$ and $\lambda \to \infty$? You do not need to give a complete proof, only an intuitive answer suffice.

Solution:

(a) It is clear that $L_{\text{ridge}}(\boldsymbol{w}) = \|X\boldsymbol{w} - \boldsymbol{y}\|^2 + \lambda\|\boldsymbol{w}\|^2$. We compute the Hessian. First, by using the chain rule the $DL(\boldsymbol{w})$ is:

$$DL(\boldsymbol{w}) = 2(X\boldsymbol{w} - \boldsymbol{y})^\top X + 2\lambda \boldsymbol{w}^\top = 2\boldsymbol{w}^\top (X^\top X + \lambda I_d) - 2\boldsymbol{y}^\top X$$

Then Hessian $D^2 L(\boldsymbol{w})$ is equal to

$$D^2 L(\boldsymbol{w}) = 2(X^\top X + \lambda I_d).$$

To prove that this matrix is positive semi-definite, we show that for any vector $\boldsymbol{v} \in \mathbb{R}^d$, we have $\boldsymbol{v}^\top D^2 L(\boldsymbol{w}) \boldsymbol{v} \geq 0$. Indeed,

$$\begin{aligned} \boldsymbol{v}^\top D^2 L(\boldsymbol{w}) \boldsymbol{v} &= 2\boldsymbol{v}^\top (X^\top X + \lambda I_d) \boldsymbol{v} \\ &= 2\boldsymbol{v}^\top X^\top X \boldsymbol{v} + 2\lambda \boldsymbol{v}^\top I_d \boldsymbol{v} \\ &= 2\|X\boldsymbol{v}\|^2 + 2\lambda \|v\|^2 \geq 0, \end{aligned}$$

simply because $\lambda \geq 0$.

(b) First, let us prove that the function f is *coercive*, i.e., $\lim_{\|x\|\to\infty} f(x) = \infty$. In the definition of strong convexity, by putting $x = 0$ we get

$$f(y) \geq f(0) + \nabla f(0)^\top y + \frac{\alpha}{2}\|y\|^2 \geq f(0) - \|\nabla f(0)\| \cdot \|y\| + \frac{\alpha}{2}\|y\|^2,$$

where we used the Cauchy-Schwartz inequality: $\nabla f(0)^\top y \geq -\|\nabla f(0)\| \cdot \|y\|$. The right-hand side of the equation above is a quadratic function of $\|y\|$ with a positive coefficient for second degree term. Thus, it goes to infinity as $\|y\| \to \infty$. Hence, f also goes to infinity.

Next, we prove that f has a global minimum. Denote by $s = \inf_{x\in\mathbb{R}^d} f(x) < \infty$. Then, by the definition of infimum, there exists a sequence $x_1, x_2, \ldots$ such that $f(x_n) \to s$. We claim that this sequence is bounded: otherwise, there was a subsequence that $\|x_{n_i}\| \to \infty$. But as f is coercive, $f(x_{n_i}) \to \infty$, contradicting $f(x_{n_i}) \to s < \infty$. Hence, the sequence $x_1, x_2, \ldots$ is inside some bounded set. By compactness, we obtain that there exists a convergent subsequence. As f is continuous, the f value of this subsequence converges as well, meaning that the infimum is attained. That is, $\exists x_\infty : f(x_\infty) = s = \inf f(x)$.

Finally, we prove uniqueness. If x and y were two distinct global minima for f, then, by strong convexity, we have

$$f\left(\frac{x+y}{2}\right) < \frac{1}{2}(f(x) + f(y)) = \min f,$$

a contradiction.

(c) By the computations of part (a), we have that

$$D^2L(w) = 2X^\top X + 2\lambda I_d.$$

By the recap session, we know that $X^\top X$ is p.s.d., hence $D^2L(w) - 2\lambda I_d \succeq 0$. That is, the ridge function is 2λ-strongly convex. By part (b) we can deduce that it has a unique minimizer if $\lambda > 0$.

(d) The term $\lambda w^\top w$ "biases" the solution towards the origin, i.e., there is a quadratic penalty for solutions w that are far from the origin. The parameter λ determines the extend of this effect: As $\lambda \to 0$, $L_{\text{ridge}}(w)$ converges to $L(w)$. As a result the optimal solution w^*_{ridge} approaches the solution of linear regression. As $\lambda \to \infty$, only the quadratic penalty $w^\top w$ is relevant and w^*_{ridge} hence approaches the null vector $(0, 0, \ldots, 0)$.

One can also show this interesting property (however, the proof is involved): Assume $n < d$ (as the situation discussed in (b)). Then w^* for linear regression is not unique. Denote by w^*_λ the *unique* solution to the Ridge regression problem for $\lambda > 0$. Then the limit $\lim_{\lambda\to 0} w^*_\lambda$ exists, and the limit point falls inside the space of solutions to linear regression problem. One can further show that this solution is the one with the minimum norm.

Exercise 3: Subgradients and the Lasso

Recall the linear regression problem, where $X \in \mathbb{R}^{n\times d}$ and $\boldsymbol{y} \in \mathbb{R}^n$. The lasso problem is a penalized linear regression formulated as

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\arg\min}\ \tfrac{1}{2}\|X\boldsymbol{w}-\boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_1. \tag{2}$$

Here, $\|\boldsymbol{w}\|_1$ is the ℓ_1 norm, defined as $\|\boldsymbol{w}\|_1 = |w_1| + \cdots + |w_d|$. The goal of this exercise is to characterize the optimal solution of (2).

It is evident that objective (2) is convex in $\boldsymbol{w}$ (if you do not see this, try to convince yourself). Hence, all local minimizers are going to be global minimizers. A very easy way to find the local minima is to set the gradient of the objective to zero. But wait. . . what is the gradient of $\|\boldsymbol{w}\|_1$? You guessed correct: at the point $\boldsymbol{0}$, $\|\cdot\|_1$ is *not* differentiable. However, one can construct a vector that *works* like a gradient, called *a subgradient*. Formally, a subgradient of a convex function $f : \mathbb{R}^d \to \mathbb{R}$ at the point $\boldsymbol{x}$ is a vector $\boldsymbol{p}$ such that for any point $\boldsymbol{z} \in \mathbb{R}^d$,

$$f(\boldsymbol{z}) \geq f(\boldsymbol{x}) + \langle \boldsymbol{p}, \boldsymbol{z}-\boldsymbol{x}\rangle. \tag{3}$$

The set of all subgradients at the point $\boldsymbol{x}$ is denoted by $\partial f(\boldsymbol{x})$. It can be shown that if f is differentiable at the point $\boldsymbol{x}$, there is only one subgradient, that is, $\partial f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$.

(a) For $f(x) = |x|$ find the subgradients at the point $x = 0$.

(b) Use the previous part to find the subgradients at the point $\boldsymbol{x} = \boldsymbol{0}$ for the function $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$.

A nice result in convex optimization tells us that $\boldsymbol{x}$ is a local minimum of the convex function f if and only if $\boldsymbol{0} \in \partial f(\boldsymbol{x})$. Observe that this is a generalization of the differentiable case.

(c) Find all the subgradients of the objective (2). Check that the optimum value $\boldsymbol{w}^\star$ of the optimization problem satisfies

$$X^\top(\boldsymbol{y} - X\boldsymbol{w}^\star) = \lambda\boldsymbol{p},$$

where $\boldsymbol{p}$ is a subgradient of $\|\boldsymbol{x}\|_1$ at the point $\boldsymbol{w}^\star$ (see (3)).

For the rest of this exercise, we treat the one-dimensional case ($d = 1$). Similar arguments can be made for the general case, but we do not cover those for simplicity. Hence, our optimization problem becomes

$$\underset{w\in\mathbb{R}}{\arg\min}\ \frac{1}{2}\sum_{i=1}^{n}(x_i w - y_i)^2 + \lambda|w|.$$

(d) With the method of subgradients and using the results you have for steps 1 and 3, find the optimal $w^\star$. Inspect your answer. Do you see a thresholding effect? Try to explain why this phenomenon happens.

(e) **(Optional)** Try to solve the problem for general $d > 1$.

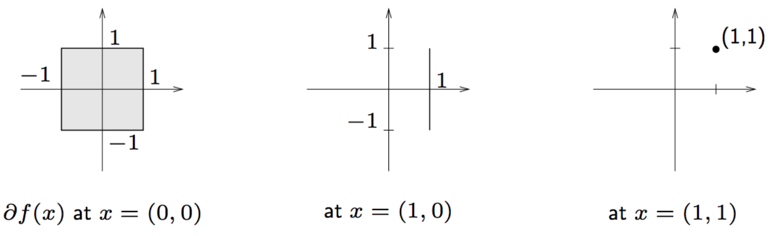

Figure 3: Subdifferential of the ℓ_1 norm for $d = 2$, computed at three different point.

Solution:

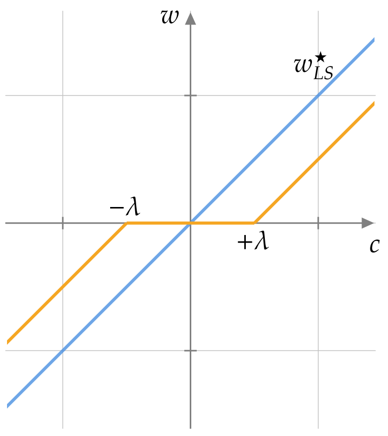

Figure 4: Least squares solution (light blue) and sparse solution induced by lasso regularization (orange).

a) For $x < 0$ the subgradient is unique and equal to the derivative $\partial f(x < 0) = \{-1\}$. The same holds for positive x, therefore $\partial f(x > 0) = \{+1\}$. At $x = 0$ the subgradient is defined by (3): $|z| \geq pz$ for all $z \in \mathbb{R}$, which is satisfied by $p \in [-1, 1]$. Therefore, $\partial f(0) = [-1, 1]$ and $\partial f(x) = \{\text{sign}(x)\}$ for all $x \neq 0$.

b) Given that $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 = \sum_{i=1}^{d} |x_i|$, and defining $f_i(\boldsymbol{x}) = x_i$, then

$$\partial f(\boldsymbol{x}) = \sum_{i=1}^{d} \partial f_i(\boldsymbol{x}) = \sum_{i:x_i \neq 0} \text{sign}(x_i)\hat{\boldsymbol{e}}_i + \sum_{j:x_j=0} [-\hat{\boldsymbol{e}}_j, \hat{\boldsymbol{e}}_j]$$

where $\hat{\boldsymbol{e}}_i$ are the coordinate system unit vectors. In Fig. 3 we show an example of how the subdifferential looks like in two dimensions.

c) By setting the subdifferential of the lasso problem in (2) to include zero, we obtain

$$\mathbf{0} \in X^{\top}(X\boldsymbol{w}^{\star} - \boldsymbol{y}) + \lambda \partial \|\boldsymbol{w}^{\star}\|_1$$

which is the same as

$$X^{\top}(\boldsymbol{y} - X\boldsymbol{w}^{\star}) = \lambda \boldsymbol{p}$$

for some $\boldsymbol{p} \in \partial \|\boldsymbol{w}^{\star}\|_1$.

d) The subdifferential of the one-dimensional problem is

$$\sum_{i=1}^{n} x_i^2 w - \sum_{i=1}^{n} x_i y_i + \lambda \partial |w| = rw - c + \lambda \partial |w|$$

which in turn is equal to

$$\partial(\text{lasso}) = \begin{cases} rw - c - \lambda & \text{if } w < 0 \\ [-c - \lambda, -c + \lambda] & \text{if } w = 0 \\ rw - c + \lambda & \text{if } w > 0 \end{cases}$$

where r is the term of the previous equation which multiplies w, and c is the second term. The optimal solution is obtained for $0 \in \partial(\text{lasso})$.

- In the first case ($w < 0$) the subgradient is zero if $w = \frac{c+\lambda}{r}$. Given that $w < 0$, this leads to the condition $c < -\lambda$.

- In the second case ($w = 0$), it is equal to zero if $0 \in [-c - \lambda, -c + \lambda]$. We can enforce this by setting that the minimum of this interval is negative and the maximum is positive, i.e. respectively $c \geq -\lambda$ and $c \leq \lambda$, leading to $-\lambda \leq c \leq \lambda$.
- The third case is similar to the first.

To sum up, we have the following system of optimal w

$$\hat{w} = \begin{cases} \frac{c+\lambda}{r} & \text{when } c < -\lambda \\ 0 & \text{when } -\lambda \leq c \leq \lambda \\ \frac{c-\lambda}{r} & \text{when } c > \lambda \end{cases}$$

which is also illustrated in Fig. 4. As you can see, the effect of λ is to set to zero all the values of $|w|$ less than λ.

e) As this question is optional, we do not provide solutions for it. Interested readers can find subgradient approaches in good resources such as Appendix 6 of the book *A Closer Look at Sparse Regression* [1].

Exercise 4: Model selection and regularization.

4.1 Validation Sets

Assume that you have access to a dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$ of $n = 10000$ data samples $(\boldsymbol{x}_i, y_i)$ that are drawn i.i.d. (independently and identically distributed) from some (unknown) distribution $p(\boldsymbol{x}, y)$.

You now need to decide how to split this dataset into a training set $\mathcal{D}_{\text{train}}$ and a validation set $\mathcal{D}_{\text{val}}$ so that you can run the following standard procedure to learn and evaluate a regression model:

Step 1: Training the regression model on $\mathcal{D}_{\text{train}}$ by minimizing the training loss

$$\widehat{f}_{\mathcal{D}_{\text{train}}} = \arg\min_{f} \left(\frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_{\text{train}}} (y_i - f(\boldsymbol{x}_i))^2 \right). \tag{4}$$

Step 2: Estimating the generalization error of the learned model $R(\widehat{f}_{\mathcal{D}_{\text{train}}})$ by computing the empirical error on $\mathcal{D}_{\text{val}}$ defined as

$$\widehat{R}_{\mathcal{D}_{\text{val}}} \left(\widehat{f}_{\mathcal{D}_{\text{train}}} \right) = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_{\text{val}}} (y_i - \widehat{f}_{\mathcal{D}_{\text{train}}}(\boldsymbol{x}_i))^2. \tag{5}$$

Remember that for a fixed estimator $f(\boldsymbol{x})$, the generalization error is defined as:

$$R(f) = \mathbb{E}_{(\boldsymbol{x}, y) \sim p}[(y - f(\boldsymbol{x}))^2].$$

[☑] Decide whether the following statements are True or False.

(a) $\widehat{R}_{\mathcal{D}_{\text{val}}}\left(\widehat{f}_{\mathcal{D}_{\text{train}}}\right)$ is more likely to provide better a estimate of the true generalisation error $R\left(\widehat{f}_{\mathcal{D}_{\text{train}}}\right)$, when using a validation set of size 500 as opposed to a validation set of size 1000.

True [✓ **False**]

Solution: The opposite should hold: with a larger validation set there will be lower variance in the estimate of generalisation error. Since the effect of this choice upon the training set size is small, it will likely have very little impact upon $\widehat{f}_{\mathcal{D}_{\text{train}}}$.

(b) Choosing a training set of size 1000 is more likely to provide a model $\widehat{f}_{\mathcal{D}_{\text{train}}}$ that has a lower true generalisation error $R\left(\widehat{f}_{\mathcal{D}_{\text{train}}}\right)$ compared to training set of size 2000.

True [✓ **False**]

Solution: The opposite holds. The reason should be intuitive. With more i.i.d data we will likely get a better predictor trained on that data.

(c) The prediction error on the training set is always less than or equal to the prediction error on the validation set, i.e.,

$$\widehat{R}_{\mathcal{D}_{\text{train}}}(\widehat{f}_{\mathcal{D}_{\text{train}}}) \leq \widehat{R}_{\mathcal{D}_{\text{val}}}\left(\widehat{f}_{\mathcal{D}_{\text{train}}}\right).$$

True [✓ **False**]

Solution: While very unlikely, it is possible for

$$\widehat{R}_{\mathcal{D}_{\text{train}}}(\widehat{f}_{\mathcal{D}_{\text{train}}}) > \widehat{R}_{\mathcal{D}_{\text{val}}}\left(\widehat{f}_{\mathcal{D}_{\text{train}}}\right)$$

. Imagine the training data has points $(1,5),(1,3)$ and the validation set only has the point $(1,4)$. $\widehat{R}_{\mathcal{D}_{\text{val}}}\left(\widehat{f}_{\mathcal{D}_{\text{train}}}\right)$ will be 0, while $\widehat{R}_{\mathcal{D}_{\text{train}}}(\widehat{f}_{\mathcal{D}_{\text{train}}})$ will be 1.

4.2 Cross-Validation

Now suppose that we used cross-validation to determine what algorithm M (e.g., SGD with which parameters) to use in order to fit f to dataset D.

[✓] Decide whether the following statements are True or False.

(a) Leave One Out Cross-validation (LOOCV) is used to estimate generalization performance of a prediction function. True [✓ **False**]

Solution: LOOCV is estimating the performance of M which is different from the generalization of a single prediction function f. For each held out validation point, LOOCV is evaluating a different f.

(b) Leave One Out Cross-validation (LOOCV) is used to compute training error. True [✓ **False**]

Solution: LOOCV is not evaluating training error but the generalization error of prediction functions learnt using M.

(c) Performing Leave One Out Cross-validation (LOOCV) on the same dataset multiple times will always yield the same result if M deterministically produces f given D. [✓ **True**] False

Solution: The validation error for each fold is deterministic given the data. The folds in LOOCV are the same for every repeat (always every data point is left out once). You can write-out the computation of the final estimate given by LOOCV and see that this statement is true, but one can also reason intuitively and see there is no source of stochasticity in this computation and therefore the same result will occur with every repeat.

4.3 Akaike Information Criterion

Define the Akaike Information Criteria as $AIC = 2k - 2ln(\hat{L})$ where $\hat{L}$ is the likelihood of f on D_{train}. k is some measure of the number of parameters used to fit f. AIC can be used for model selection by selecting models with low AIC score.

[✓] Decide whether the following statements are True or False.

(a) AIC is an estimator used to compute generalization error. ☐ True [✓ **False**]

Solution: AIC is not an estimate of generalisation error. Firstly, it is computed on D_{train} (all estimates of generalisation error that we've seen use validation datasets), and secondly it uses a combination of k and the likelihood that one would have no reason to believe would relate to directly estimating generalisation error. It is computing a score for each model based on how likely it is to be the true model, but this score is not an estimate of generalisation error.

(b) AIC encourages overfitting. ☐ True [✓ **False**]

Solution: The term based on k is explicitly encouraging models with fewer parameters, and models with fewer parameters are less able to overfit data.

4.4 Regularization

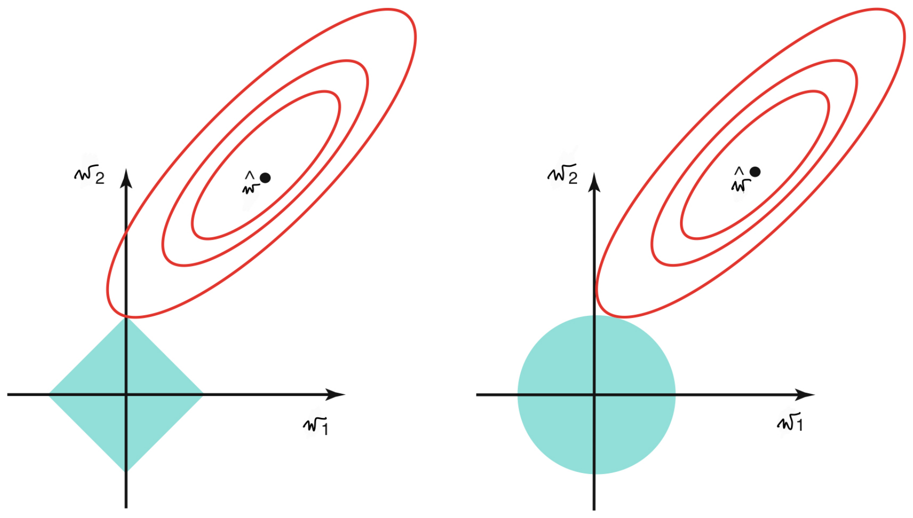

Figure 5: Blue diamond (left) and circle (right) represent constraints for the weights of f. Red curves represent contours for the loss function. That is, points with the same loss value lie on the same red curve. $\hat{w}$ achieves the minimum training loss for the unconstrained problem. The solution to the constrained regression problem is pictured as the intersection of the contour with the constraint set, since this is the lowest loss achievable within the constraint set.

Here instead of computing a prediction function based upon $\sum_{x,y \in D_{\text{train}}} l(f(x), y) + \lambda \text{Reg}(f)$ where Reg is a regularizer, we do regression by optimizing $\sum_{x,y \in D_{\text{train}}} l(f(x), y)$ subject to the constraint $\text{Reg}(f) = \eta$.

[✓] Decide whether the following statements are True or False.

(a) Choose which plot in Figure 5 corresponds to the ridge regression

1. left **2. right**

Solution: Plot the constraint $\|w\|_2 = \eta$ in the two dimensional plane.

(b) Choose which plot in Figure 5 corresponds to the lasso regression

1. left 2. right

Solution: Plot the constraint $\|w\|_1 = \eta$ in the two dimensional plane.

Exercise 5: Support Vector Machines

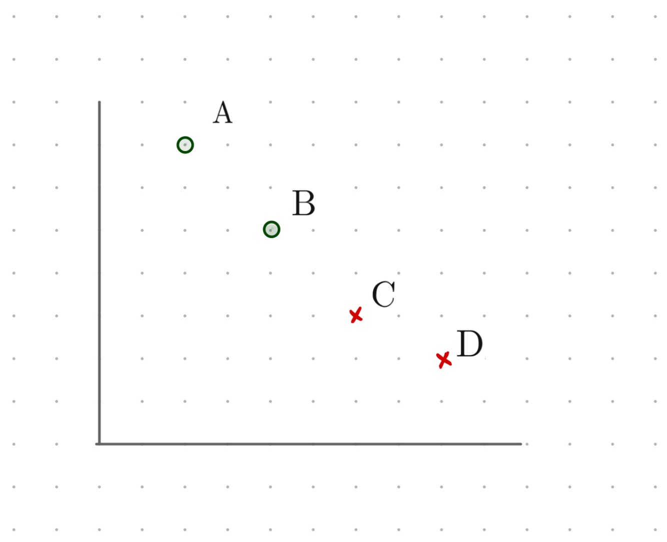

Figure 6: The four points A = (1, 3.5), B = (2, 2.5), C = (3, 1.5), D = (4, 1) are from two different classes.

Figure 6 illustrates a training dataset with of 4 points A, B, C, D where 1 or -1 indicates the class to which the point belongs. The data is linearly separable and we want to find maximum margin hyperplane that would divide the points.

(a) [✓] Derive the equation $w_1x_1 + w_2x_2 + b = 0$ for the hyperplane corresponding to the maximum margin classifier.

Solution: $x_1 - x_2 - 0.5 = 0$

(b) [✓] Compute the margin of the maximum margin classifier to two decimal places.

Solution: $1/\sqrt{2}$ (0.71)

(c) Graphically illustrate a binary classification dataset which, with hard SVM, would have the maximum possible number of support vectors in each class.

Solution: All points lay on a line.

(d) Consider a general setup with input $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where $y_i = 1$ or $y_i = -1$, and the data is linearly separable. As you have computed above for 2D case, maximum margin classifier aims at a hyperplane that (a) separates the two classes of data and (b) the distance between them is as large as possible. Now, write down the objective of maximum margin classifier in terms of $\|\boldsymbol{w}\|$. What is the relation between the value of the margin and $\|\boldsymbol{w}\|$?

Solution: As discussed in more detail in the tutorial, the margin $M(\boldsymbol{w}) = \frac{1}{||\boldsymbol{w}||}$. Let's briefly recap that:

Remember, that for the support vectors $y_i\boldsymbol{w}^T\boldsymbol{x}_i = 1$. Let's choose two support vectors $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ from the two different classes. Then, the margin is the distance between $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ projected to the normal $\boldsymbol{w}$:

$$2M(\boldsymbol{w}) = \frac{\boldsymbol{w}^T(\boldsymbol{x}_+ - \boldsymbol{x}_-)}{\|\boldsymbol{w}\|} = \frac{\boldsymbol{w}^T\boldsymbol{x}_+ - \boldsymbol{w}^T\boldsymbol{x}_-}{\|\boldsymbol{w}\|} = \frac{1-(-1)}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}.$$

(e) In the tutorial, we discussed soft SVM and how to compute subgradients with respect to the objective used. Write a pseudo-code implementing SGD for soft SVM. The inputs are the data points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, regularization constant λ, batch size b, learning rates $\{\eta_t\}_{t\in\mathbb{N}}$, and number of iterations T. Explicitly write the loss you are minimizing and the closed form of the gradients you are computing at each descent step.

Solution: Before writing the algorithm, we compute the (sub)gradient of the hinge loss at any point. Define $\ell_{\text{hinge}}(z) = \max\{0, 1-z\}$. The loss for the datapoint $(\boldsymbol{x}^i, y_i)$ and the hyperplane $\boldsymbol{w}$ evaluates as $\ell_{\text{hinge}}(y_i\boldsymbol{w}^\top\boldsymbol{x}^i)$. The (sub)gradient can be found as follows:

$$\nabla_{\boldsymbol{w}}\ell_{\text{hinge}}(y_i\boldsymbol{w}^\top\boldsymbol{x}^i) = \begin{cases} \boldsymbol{0} & \text{if } \; y_i\boldsymbol{w}^\top\boldsymbol{x}^i \geq 1 \\ -y_i\boldsymbol{x}^i & \text{otherwise.} \end{cases}$$

Note, that as the function ℓ_{hinge} is not differentiable at 1, we have to choose a subgradient at that point. Moreover, we abuse the notation and denote both cases by the symbol ∇.

Now we can write the algorithm:

```
Input: $\{(\boldsymbol{x}^i, y_i)\}_{i=1,\dots,N}$, $\lambda$, $b$, $T$, $\{\eta_t\}_{t\in\mathbb{N}}$
Output: A hyperplane $\boldsymbol{w}_{\text{svm}}$
Set $\boldsymbol{w}^1 \leftarrow \boldsymbol{0}$;
for $t = 1, 2, \dots, T$ do
    Draw $b$ samples from the dataset uniformly at random with replacement, calling them
      $\{\boldsymbol{x}^{(i)}, y_{(i)}\}_{i\leq b}$ ;
    Let $\boldsymbol{g} = \lambda\boldsymbol{w}^t$ ;          // $\boldsymbol{g}$ will hold the gradient at this iteration
    for $i = 1, \dots, b$ do
        if $y_{(i)}(\boldsymbol{w}^t)^\top\boldsymbol{x}^{(i)} < 1$ then
            $\boldsymbol{g} \leftarrow \boldsymbol{g} - y_{(i)}\boldsymbol{x}^{(i)}$ ;
        end
    end
    $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \eta_t\boldsymbol{g}$ ;          // The gradient descent step
end
return $\boldsymbol{w}^{T+1}$
```

Algorithm 1: SGD on SVM