ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Machine Learning – CS-433

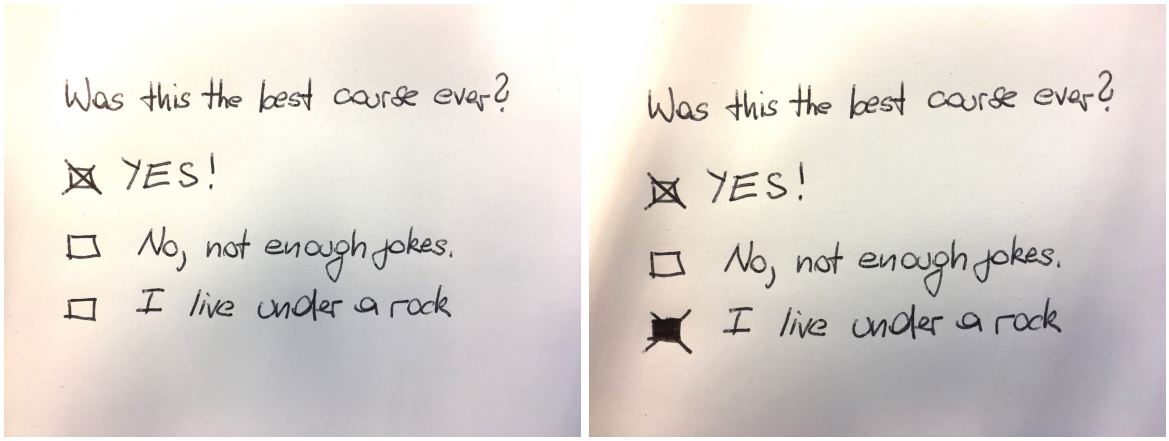
January 17, 2019

16h15-19h15 (STCC08328)

Important Notes

- This exam counts for 60 percent towards your final grade.
- This is a closed book exam. No electronic devices of any kind.
- Place on your desk: your student ID, writing utensils, one double-sided A4 page crib sheet (handwritten or 11pt min font size) if you have one; place all other personal items below your desk or on the side.
- **Mark your answer with a thick 'X' in the corresponding box (see example below)**. If you want to change your answer, fill the box completely and mark the new answer with an 'X'. If you are unsure about an answer, perhaps just mark your best initial guess somewhere on the side before inserting the 'X' later on.
- Each page has a code on top of it (see the top of this page). Do not write on it.
- For the MCQ part, more than one answer can be correct. There are negative points for incorrect answers.
- You each have a different exam.
- For technical reasons, **do not use pencils for the MCQ part, use only pens**.

  Good luck!



The left figure shows how you mark an answer correctly.
. This part will be graded automatically and if the line width is not sufficiently large, the system will not register your answer correctly.
The figure on the right shows how you can "erase" an answer.
Completely fill in the box corresponding to your old answer and then mark your new answer.

Room: **STCC**
Seat: **1**

Student Name / Sciper no.:
I Know All
3941009

Problem 1 [1.8 points]

Let $f(x, y)$ be a general function over $\mathbb{R}^2$. Mark any of the following statements that is *always* (independent of the function) correct?

- ☐ $\max_{y'} \min_{x'} f(x', y') \leq \min_{x'} \max_{y'} f(x', y')$
- ☐ $\min_{x'} f(x', y) \leq \min_{y'} f(x, y'), \forall x, y$
- ☐ $\max_{x'} f(x', y) \leq \max_{y'} f(x, y'), \forall x, y$
- ☐ $\min_{x'} f(x', y) \leq \max_{y'} f(x, y'), \forall x, y$
- ☐ $\min_{x'} f(x', y) \leq \max_{x'} f(x', y), \forall y$
- ☐ $\min_{y'} \max_{x'} f(x', y') \leq \max_{x'} \min_{y'} f(x', y')$

Solution: The correct choices are: $\min_{x'} f(x', y) \leq \max_{x'} f(x', y), \forall y$, $\min_{x'} f(x', y) \leq \max_{y'} f(x, y'), \forall x, y$, $\max_{y'} \min_{x'} f(x', y') \leq \min_{x'} \max_{y'} f(x', y')$.

Problem 2 [1 point]

You are performing classification in d dimensions (where d is very large) using a k-nearest neighbor classifier. You have N training samples and get an acceptable performance. Your boss hands you a similar classification task but in $2d$ dimensions. How many training samples will you need to get a comparable performance? Make an educated guess.

- ☐ $2N$
- ☐ e^N
- ☐ N^2
- ☐ $\sqrt{d}$
- ☐ e^d
- ☐ $\sqrt{N}$
- ☐ $\log(N)$
- ☐ d^2
- ☐ $2d$
- ☐ $\log(d)$

Solution:
Recall that the amount of data we need in order to "cover" the space grows roughly exponentially in the dimension. Therefore, if we are working in twice the dimensions, then we should need the square of the amount of data.

Problem 3 [1 point]

Consider the following joint distribution on X and Y, where $X \in \{-1, 0, 1\}$ and $Y \in \{0, 1\}$: $p(X = -1, Y = 0) = 0.05$, $p(X = -1, Y = 1) = 0.05$, $p(X = 0, Y = 0) = 0.1$, $p(X = 0, Y = 1) = 0.1$, $p(X = 1, Y = 0) = 0.3$, $p(X = 1, Y = 1) = 0.4$. You learn that $X \geq 0$. What is the largest probability of being correct you can achieve when predicting Y in this case?

- ☐ $\frac{5}{9}$
- ☐ 1
- ☐ $\frac{2}{3}$
- ☐ $\frac{1}{4}$
- ☐ $\frac{1}{2}$
- ☐ $\frac{1}{7}$
- ☐ $\frac{1}{3}$
- ☐ $\frac{6}{7}$
- ☐ 0
- ☐ $\frac{4}{7}$
- ☐ $\frac{3}{7}$

Solution: We have

$$\begin{aligned} p(Y = 1 | X \geq 0) &= \frac{p(X \geq 0, Y = 1)}{p(X \geq 0)} \\ &= \frac{p(X \geq 0, Y = 1)}{1 - p(X = -1)} \\ &= \frac{p(X = 0, Y = 1) + p(X = 1, Y = 1)}{1 - p(X = -1, Y = 0) + p(X = -1, Y = 1)} \\ &= \frac{0.1 + 0.4}{1 - 0.1} = \frac{5}{9}. \end{aligned}$$

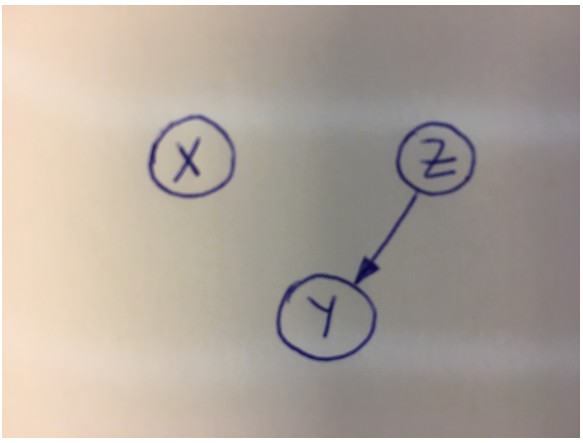
Therefore, you should guess that $Y = 1$ and you will be right a fraction $5/9$ of the time. This is the best you can do.

Problem 4 [1.8 point]

You are given a Bayes net involving the random variables X, Y, and Z of the form $X \to Y \leftarrow Z$. What conclusions can you draw? [Recall that $\perp$ means independent and $|\cdots$ means conditioned on $\cdots$.]

- ☐ $X \perp Y \quad | Z$
- ☐ $X \perp Z$
- ☐ $X \perp Z \quad | Y$
- ☐ $X \perp Y$
- ☐ $Y \perp Z \quad | X$
- ☐ $Y \perp Z$

Solution: The Bayes net implies a factorization $p(x, y, z) = p(x)p(z)p(y|x, z)$. From this we see that $X \perp Z$, and this is the only conclusion that can be drawn.



Problem 5 [2.5 points]

Consider the directed acyclic graph shown above. Which of the following factorizations of the random variables X, Y and Z are consistent with this graph. We say that a factorization is *consistent* with a Bayes net if *every* distribution characterized by the Bayes net can be written in the form of this factorization.

- ☐ $p(X, Y, Z)$
- ☐ $p(X)p(Z)p(Y|Z)$
- ☐ $p(X, Z)p(Y|X, Z)$
- ☐ $p(X)p(Y)p(Z)$
- ☐ $p(X)p(Z)p(Y|X)$
- ☐ $p(X)p(Z|X)p(Y|X, Z)$
- ☐ $p(X)p(Y)p(Z|Y)$
- ☐ $p(X)p(Y|X)p(Z)$
- ☐ $p(Y)p(X, Z)$
- ☐ $p(X)p(Z)p(Y|X, Z)$

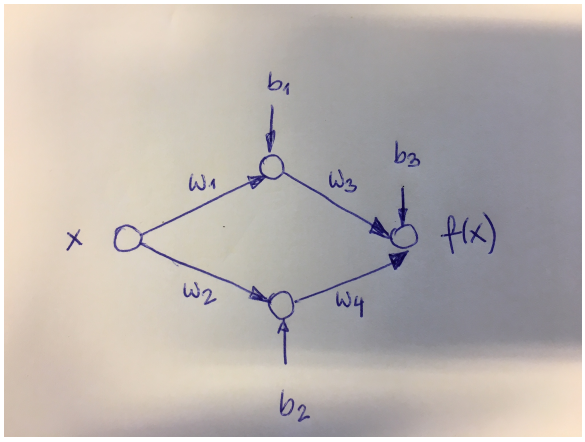
Solution: The following factorizations are consistent: $p(X)p(Z)p(Y|Z)$, $p(X)p(Y)p(Z|Y)$, $p(X)p(Z \mid X)p(Y|X, Z)$, $p(X)p(Z)p(Y \mid X, Z)$, $p(X, Z)p(Y \mid X, Z)$, $p(X, Y, Z)$

Problem 6 [2.4 points]

Which of the following statements are correct?

- ☐ One iteration of standard SGD for SVM costs roughly $\Theta(D)$, where D is the dimension.
- ☐ Unions of convex sets are convex.
- ☐ Hinge loss (as in SVMs) is typically preferred over L2 loss (least squares loss) in classification tasks.
- ☐ In PCA, the first principal direction is the eigenvector of the data matrix $\boldsymbol{X}$ with largest associated eigenvalue.
- ☐ MSE (mean squared error) is typically more sensitive to outliers than MAE (mean absolute error).
- ☐ One iteration of standard SGD for logistic regression costs roughly $\Theta(ND)$, where N is the number of samples and D is the dimension.

Solution: Hinge loss (as in SVMs) is typically preferred over L2 loss (least squares loss) in classification tasks.
MSE (mean squared error) is typically more sensitive to outliers than MAE (mean absolute error)
One iteration of standard SGD for SVM costs roughly $\Theta(D)$, where D is the dimension

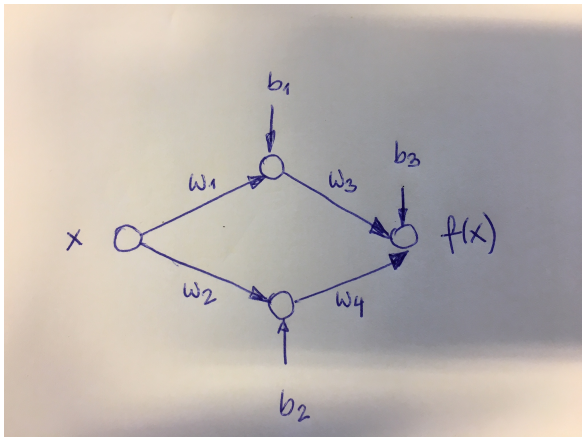


Problem 7 [1 point]

Consider the simple neural network shown above and let $\mathcal{L}$ be the objective function (this function depends on the parameters $\Theta = (w_1, w_2, w_3, w_4, b_1, b_2)$, the activation functions, and the loss function at the output).
For the given input-output pair $(\tilde{x}_n, \tilde{y}_n)$ we compute the gradient wrt Θ. We get the result, $\frac{\partial \mathcal{L}}{\partial w_1} = 1$, $\frac{\partial \mathcal{L}}{\partial w_2} = 1$, $\frac{\partial \mathcal{L}}{\partial w_3} = 1$, $\frac{\partial \mathcal{L}}{\partial w_4} = 1$, $\frac{\partial \mathcal{L}}{\partial b_1} = 1$, and $\frac{\partial \mathcal{L}}{\partial b_2} = 1$.
Assume now that we have some weight sharing, in particular assume that $w_2 = w_3$, and that for the above Θ this condition was already fulfilled. What is $\frac{\partial \mathcal{L}}{\partial w_1}$?

- ☐ 2
- ☐ 0
- ☐ 1
- ☐ 3
- ☐ not sufficient information to determine

Solution: Since no other parameter shares the weight with w_1, we have $\frac{\partial \mathcal{L}}{\partial w_1} = 1$.



Problem 8 [1 point]

Consider once more the scenario above with the same input-output pair $(\tilde{x}_n, \tilde{y}_n)$ and the same Θ except that the edge with weight w_3 is missing (and so Θ does not contain w_3). There is no weight sharing. What is $\frac{\partial \mathcal{L}}{\partial w_1}$?

- ☐ 1
- ☐ 0
- ☐ 3
- ☐ 2
- ☐ not sufficient information to determine

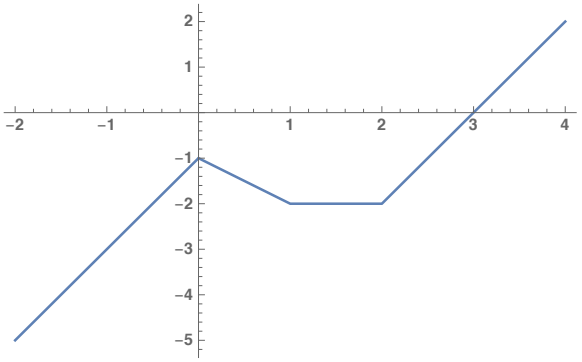
Solution: We have $\frac{\partial \mathcal{L}}{\partial w_1} = 0$ since there is no connection from this edge to the output.

Problem 9 [1 point]

Consider once more the scenario above with the same input-output pair $(\tilde{x}_n, \tilde{y}_n)$ and the same Θ except that the edge with weight w_4 is missing (and so Θ does not contain w_4). There is no weight sharing. What is $\frac{\partial \mathcal{L}}{\partial w_1}$?

- ☐ not sufficient information to determine
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 0

Solution: Due to the missing edge with weight w_4 the operating point is changing (forward pass in the backpropgation algorithm). This changes also all the partial derivatives and we do not have sufficient information to determine how. NOTE: This problem caused some confusion.



Problem 10 [1.5 point]

Mark all x-values for which the shown function $f(x)$ has a subgradient?

- ☐ 1
- ☐ 0
- ☐ 2
- ☐ 3
- ☐ -1

Solution: At $x = 2$ and $x = 3$ the function has a subgradient and we can assign it the value 2.

Problem 11 [1.5 point]

Consider our standard least-squares problem

$$\text{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \text{argmin}_{\mathbf{w}} \frac{1}{2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \frac{\lambda}{2} \sum_{d=1}^{D} w_d^2.$$

Here, $\{(\mathbf{x}_n y_n)\}_{n=1}^N$ is the data. The N-length vector of outputs is denoted by $\mathbf{y}$. The $N \times D$ data matrix is called $\mathbf{X}$. It's rows contain the tuples $\mathbf{x}_n$. Finally, the parameter vector of length D is called $\mathbf{w}$. (All just like we defined in the course). Mark any of the following formulas that represent an equivalent way of solving this problem.

- ☐ $\text{argmin}_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_N) \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{y}$
- ☐ $\text{argmin}_{\mathbf{w}} \sum_{n=1}^{N} [1 - y_n \mathbf{x}_n^\top \mathbf{w}]_+ + \frac{\lambda}{2} \|\mathbf{w}\|^2$. Recall: $[z]_+ = \max\{0, z\}$
- ☐ $\text{argmin}_{\mathbf{w}} - \log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) p(\mathbf{w})$, where $p(\mathbf{w})$ correspond to the density of a D-length vector of iid zero-mean Gaussians with variance $1/\lambda$ and $p(\mathbf{y} \mid \mathbf{X}, \mathbf{w})$ corresponds to the density of a vector of length N of independent Gaussians of mean $\mathbf{x}_n^\top \mathbf{w}$, variance 1 and observation $\mathbf{y}_n$ for component n.
- ☐ $\text{argmin}_{\mathbf{w}} \frac{1}{2} \sum_{n=1}^{N} \ln(1 + e^{\mathbf{x}_n^\top \mathbf{w}}) - y_n \mathbf{x}_n^\top \mathbf{w}$
- ☐ $\text{argmin}_{\mathbf{w}} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$

Solution: The following are correct:

$$\text{argmin}_{\mathbf{w}} - \log p(\mathbf{y} \| \mathbf{X}, \mathbf{w}) p(\mathbf{w}),$$
$$\text{argmin}_{\mathbf{w}} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2,$$
$$\text{argmin}_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_N) \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{y}.$$

Problem 12 [1 point]

Which of the following statements about the SVD of an $N \times D$ matrix $\mathbf{X}$ are correct?

- ☐ We can compute the singular values of $\mathbf{X}$ by computing the eigenvalues of $\mathbf{X}\mathbf{X}^\top$. This has complexity $O(N^3)$.
- ☐ We can compute the singular values of $\mathbf{X}$ by computing the eigenvalues of $\mathbf{X}\mathbf{X}^\top$. This has complexity $O(D^3)$.
- ☐ We can compute the singular values of $\mathbf{X}$ by computing the eigenvalues of $\mathbf{X}^\top\mathbf{X}$. This has complexity $O(N^3)$.
- ☐ We can compute the singular values of $\mathbf{X}$ by computing the eigenvalues of $\mathbf{X}^\top\mathbf{X}$. This has complexity $O(D^3)$.
- ☐ We can compute the singular values of $\mathbf{X}$ by computing the eigenvalues of $\mathbf{X}\mathbf{X}^\top$ if only if $\mathbf{X}$ is a square matrix. This has complexity $O(D^3) = O(N^3)$.

Solution: We can compute the singular values of $\mathbf{X}$ by computing the eigenvalues of $\mathbf{X}^\top\mathbf{X}$. This has complexity $O(D^3)$. We can compute the singular values of $\mathbf{X}$ by computing the eigenvalues of $\mathbf{X}\mathbf{X}^\top$. This has complexity $O(N^3)$.

Problem 13 [1 point]

Assume that you have given linearly separable data for a classification task. You are using support vector machines. More precisely, you are optimizing

$$\text{argmin}_{\mathbf{w}} \sum_{n=1}^{N} [1 - y_n \mathbf{x}_n^\top \mathbf{w}]_+ + \frac{\lambda}{2} \|\mathbf{w}\|^2,$$

with a very large value of λ. Recall: $[z]_+ = \max\{0, z\}$. Assume that $(\tilde{\mathbf{x}}_n, \tilde{y}_n)$ is an essential support vector, i.e., a data point that lies exactly on the margin. What is the *most likely* outcome if you remove this support vector from the data and rerun the algorithm with the remaining $N - 1$ data points?

- ☐ I will get exactly the same outcome and the same margin.
- ☐ The optimal vector $\mathbf{w}^*$ will change and the margin will double.
- ☐ The optimal vector $\mathbf{w}^*$ will change and the margin will decrease.
- ☐ The optimal vector $\mathbf{w}^*$ will change and the margin will increase.
- ☐ The algorithm will not work since the data is missing an essential support vector.

Solution: The optimal vector $\mathbf{w}^*$ will change and the margin will increase.

Problem 14 [1.8 point]

Consider a regression task. You are using your favorite learning algorithm with parameters $\mathbf{w}$ and add a regularization term of the form $\frac{\lambda}{2}\|\mathbf{w}\|^2$. Which of the following statements are correct for a typical scenario?

- ☐ The training error as a function of $\lambda \geq 0$ decreases.
- ☐ The training error as a function of $\lambda \geq 0$ increases.
- ☐ The test error as a function of $\lambda \geq 0$ increases.
- ☐ The test error as a function of $\lambda \geq 0$ decreases.
- ☐ The training error as a function of $\lambda \geq 0$ first decreases and then increases.
- ☐ The test error as a function of $\lambda \geq 0$ first decreases and then increases.

Solution: The test error as a function of $\lambda \geq 0$ first decreases and then increases. The training error as a function of $\lambda \geq 0$ increase.

Problem 15 [1.8 points]

Consider optimizing a matrix factorization $\boldsymbol{W}\boldsymbol{Z}^\top$ in the matrix completion setting, for $\boldsymbol{W} \in \mathbb{R}^{D\times K}$ and $\boldsymbol{Z} \in \mathbb{R}^{N\times K}$. We write Ω for the set of observed matrix entries. Which of the following statements are correct?

- ☐ Given any Ω, for $K := \min\{N, D\}$, there is an exact solution to the problem.
- ☐ In general, a step of SGD will change all entries of the $\mathbf{W}$ and $\mathbf{Z}$ matrices.
- ☐ Adding a Frob-norm regularizer for $\boldsymbol{W}$ and $\boldsymbol{Z}$ to the matrix factorization objective function makes the objective convex.
- ☐ A step of alternating least squares is more costly than an SGD step.
- ☐ For complete observations $\Omega = [1 \dots D] \times [1 \dots N]$, the problem can be solved by the singular value decomposition.
- ☐ The cost of an SGD step depends on the number of observed entries.

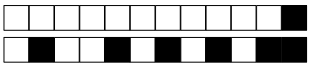
Solution: A step of alternating least squares is more costly than an SGD step. Given any Ω, for $K := \min\{N, D\}$, there is an exact solution to the problem. For complete observations $\Omega = [1 \dots D] \times [1 \dots N]$, the problem can be solved by the singular value decomposition.

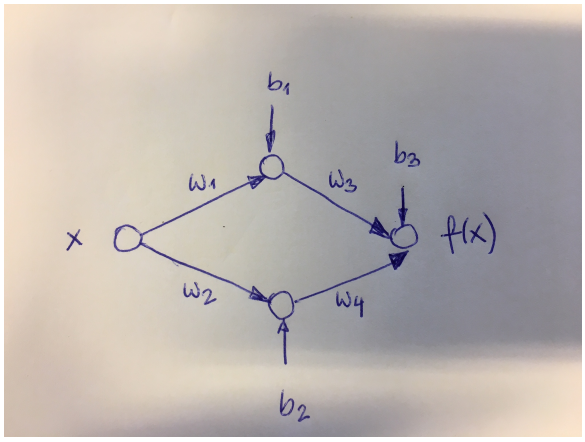
Problem 16 [2 points]

In Text Representation learning, which of the following statements are correct?

- ☐ Learning GloVe word vectors can be done using the singular value decomposition, if the f_{dn} weights are set to 1 for all observed entries.
- ☐ The skip-gram model for learning original word2vec embeddings does learn a binary classifier for each word.
- ☐ FastText as discussed in the course learns word vectors and sentence representations which are specific to a supervised classification task.
- ☐ Logistic regression used for text classification is faster at *test time* when using word vectors as opposed to bag-of-word representation of the input.

Solution: The skip-gram model for learning original word2vec embeddings does learn a binary classifier for each word. FastText as discussed in the course learns word vectors and sentence representations which are specific to a supervised classification task.





Problem 17 [4 points]

Consider the simple neural net with one hidden layer and sigmoid activation functions. Assume that you want to approximate the following function $f(x)$ for $x \in [0, 1]$:

$$f(x) = 2\mathbb{1}_{\{0 \le x < \frac{1}{3}\}} - \mathbb{1}_{\{\frac{1}{3} \le x < \frac{2}{3}\}}.$$

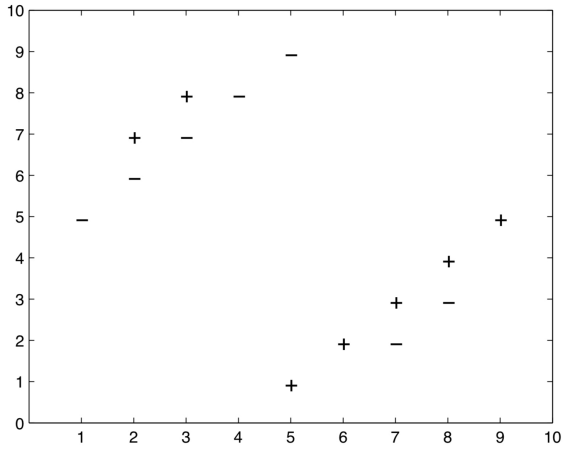
Find a choice of the parameters $w_1, w_2, w_3, w_4, b_1, b_2, b_3$ so that you get a good approximation of $f(x)$.
Recall: $\mathbb{1}_{\{\dots\}}$ is the function that is 1 whenever the condition within the curly brackets is fulfilled and 0 otherwise.

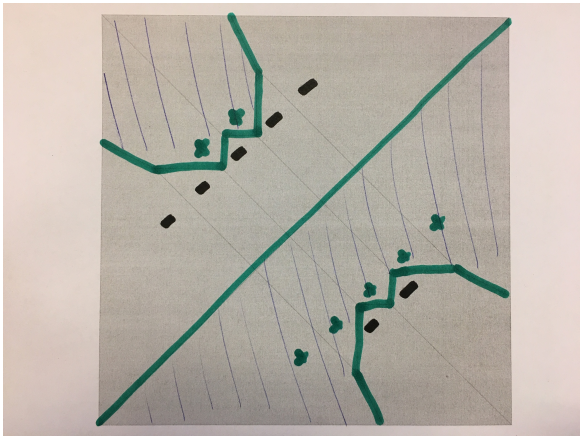
☐0 ☐0.5 ☐1 ☐1.5 ☐2 ☐2.5 ☐3 ☐3.5 ☐4 *Reserved, do not write on this box*

Solution: The first important point to realize is that the given network can only realize two jumps. Luckily this is all you need since you are only asked to approximate the function on $[0, 1]$. Hence you only have to model the jumps at $x = \frac{1}{3}$ and $x = \frac{2}{3}$. There is no need to model the "jump" at $x = 0$. The second important point is to realize that the weights w_1 and w_2 have to be taken large in magnitude in order to create "sharp" transitions.
There are four distinct ways of approximating this function. These are

- $f(x) = 2 - 3\mathbb{1}_{\{x \ge \frac{1}{3}\}} + \mathbb{1}_{\{x \ge \frac{2}{3}\}}$
- $f(x) = 3\mathbb{1}_{\{x \le \frac{1}{3}\}} - \mathbb{1}_{\{x \le \frac{2}{3}\}}$
- $f(x) = -1 + 3\mathbb{1}_{\{x \le \frac{1}{3}\}} + \mathbb{1}_{\{x \ge \frac{2}{3}\}}$
- $f(x) = 3 - \mathbb{1}_{\{x \le \frac{2}{3}\}} - 3\mathbb{1}_{\{x \ge \frac{1}{3}\}}$

For e.g., the first approach this results in the following choices: w_1 and w_2 positive and very large. Let $b_1 = -\frac{1}{3}w_1$, $b_2 = -\frac{2}{3}w_2$, $w_3 = -2$, $w_4 = 1$, and $b_3 = 2$. Of course, you can also exchange the roles of the top and the bottom branch.





Problem 18 [4 points]
(Due to C. Guestrin) Consider a k-nearest neighbor classifier using the Euclidean metric for a binary classification task. Recall, that given k (think of k as odd), for any point x we look at the labels of the k nearest neighbors and assign to this point the label corresponding to the majority of these labels. Note that a point can be its own neighbor. Consider the training data shown.

1. What value of k minimizes the training set error for this dataset? What is the resulting training error?
2. Why might using relatively large values of k be bad in this dataset?
3. What value of k minimizes leave-one-out cross-validation error for this dataset? What is the resulting error? NOTE: Recall that in the leave-one-out cross-validation procedure we proceed as follows. Assume we have given N data points. We drop one data point at a time and predict the label for this dropped data point based on the remaining $N-1$ data points. We do this for each of the N data points and the error is the average of the error for each of the N cases.
4. In the figure itself, sketch the 1-nearest neighbor decision boundary for this dataset.

☐0 ☐0.5 ☐1 ☐1.5 ☐2 ☐2.5 ☐3 ☐3.5 ☐4 *Reserved, do not write on this box*

Solution:

1. For $k = 1$ we get a training error of 0. NOTE: Training error refers to the error you get if you apply the learned function on the training data. Hence it is irrelevant that k-nearest neighbor does not have a traditional "training" phase.
2. If we are using relatively large values of k the model will become very simple (constant once we choose k close to N) and we will severly underfit.
3. The best choice is $k = 5$ or $k = 7$ (assuming that we choose it odd). For this choice we will make a mistake for the 4 points at the extreme boundary and will get the correct prediction for the 10 points closer to the center. Hence the error is $2/7$.
4. See the plot. Many people got the regions approximately right but not exactly right. The regions are piece-wise linear.

Many people more or less solved this problem correctly. Almost nobody got the region in point 4 exactly right but we deduced only few points in this case if any.

Problem 19 [4 points]

In the following let $\kappa_1(\mathbf{x}, \mathbf{x}')$ and $\kappa_2(\mathbf{x}, \mathbf{x}')$ be two valid kernels. Show that the following are also valid kernels:

1. $\kappa(\mathbf{x}, \mathbf{x}') = a\kappa_1(\mathbf{x}, \mathbf{x}') + b\kappa_2(\mathbf{x}, \mathbf{x}')$ for all $a, b \geq 0$.
2. $\kappa(\mathbf{x}, \mathbf{x}') = \kappa_1(\mathbf{x}, \mathbf{x}')\kappa_2(\mathbf{x}, \mathbf{x}')$.
3. $\kappa(\mathbf{x}, \mathbf{x}') = \kappa_1(f(\mathbf{x}), f(\mathbf{x}'))$, where f is any function from the domain to itself.
4. $\kappa(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})f(\mathbf{x}')$ for any real-valued function f.

☐0 ☐0.5 ☐1 ☐1.5 ☐2 ☐2.5 ☐3 ☐3.5 ☐4 *Reserved, do not write on this box*

Solution:

1. By assumption κ_1 and κ_2 are valid kernels. Hence there exist feature maps ϕ_1 and ϕ_2 so that

$$\kappa_1(\mathbf{x}, \mathbf{x}') = \phi_1(\mathbf{x})^\top \phi_1(\mathbf{x}'),$$
$$\kappa_2(\mathbf{x}, \mathbf{x}') = \phi_2(\mathbf{x})^\top \phi_2(\mathbf{x}').$$

   Hence,

$$\kappa(\mathbf{x}, \mathbf{x}') = a\phi_1(\mathbf{x})^\top \phi_1(\mathbf{x}') + b\phi_2(\mathbf{x})^\top \phi_2(\mathbf{x}').$$

   This can be represented as an inner product via the feature map

$$(\sqrt{a}\phi_1(\cdot), \sqrt{b}\phi_2(\cdot)).$$

2. Let the two feature maps be ϕ_1 and ϕ_2. Assume that they are of dimensions d_1 and d_2. Then ϕ is a feature map of dimension $d_1 d_2$ of the form

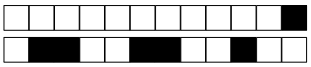
$$\begin{aligned}\phi(\mathbf{x})^\top =&((\phi_1(\mathbf{x}))_1(\phi_2(\mathbf{x}))_1, \cdots, (\phi_1(\mathbf{x}))_1(\phi_2(\mathbf{x}))_{d_2}, \cdots, \\ &(\phi_1(\mathbf{x}))_{d_1}(\phi_2(\mathbf{x}))_1, \cdots, (\phi_1(\mathbf{x}))_{d_1}(\phi_2(\mathbf{x}))_{d_2}).\end{aligned}$$

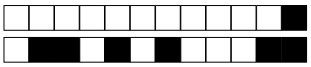
3. Let $\phi_1(\cdot)$ be the feature map corresponding to $\kappa_1(\cdot, \cdot)$. Then by direct inspection we see that $\phi(\cdot) = \phi_1(f(\cdot))$ is the feature map corresponding to $\kappa(f(\cdot), f(\cdot))$. Indeed,

$$\phi_1(f(\mathbf{x}))^\top \phi_1(f(\mathbf{x}')) = \kappa_1(f(\mathbf{x}), f(\mathbf{x}') = \kappa(\mathbf{x}, \mathbf{x}').$$

4. Clearly, $\phi(\mathbf{x}) = f(\mathbf{x})$ will be the corresponding feature map.

This problem caused lots of confusion and only few people got full points. Most people did not take the "easy" route outlined above but stated Mercer's condition. Recall that if we are given a $\kappa(\cdot, \cdot)$ and a set of points $\{\mathbf{x}_n\}_{n=1}^N$ then we can associate to this an $N \times N$ matrix $\mathbf{K}$ with entries $\mathbf{K}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. Mercer's condition then requires that this matrix be symmetric and psd for any given set of points. Only few people stated what $\mathbf{K}$ was with respect to κ. The symmetry was proven correctly by most but almost nobody gave a correct proof of the psd property. E.g., the product of two psd matrices is in general *not* psd. Adding to the confusion, some people first stated the correct representation as an inner product and *then* still tried to prove that Mercer's condition was fulfilled. Recall, Mercer's condition guarantees that κ can be written as an inner product. But once you have explicitly provided an inner product representation there is no point in checking Mercer's condition. Other frequently made assertions: For point 4 many people stated that since the function f maps points from the domain to itself the kernel was "still" valid. Indeed, this is what was asked to prove and the proof is simple. E.g., if you go the psd route you can observe that the matrix is simply now a matrix corresponding to different choices of points and that the original matrix had to be psd for *any* choice of points. But simply stating that it was just a valid kernel did not get you any points.





Problem 20 [4 points]
Consider the k-means algorithm. We discussed in the course that this algorithm is efficient. But we also discussed that it might not converge to the optimal solution. Let us explore this in a very simple setting.
Assume that your data is one-dimensional. I.e., the points of your training set S_{training} are elements of $\mathbb{R}$. Further, assume that $k = 2$, i.e., we are looking for two clusters.
Give an example of a data set in one dimension that has at least two distinct fixed points. I.e., a data set so that depending on the initial choice of cluster assignments the algorithm will converge to different solutions. The simpler the example the better (and the more points).

☐0 ☐0.5 ☐1 ☐1.5 ☐2 ☐2.5 ☐3 ☐3.5 ☐4 *Reserved, do not write on this box*

Solution: Let $S_{\text{training}} = \{-1, 0, 1\}$. If our initial cluster assignments are $\{-1, 0\}$ for cluster 1 and $\{1\}$ for cluster 2 then this itself is already a fixed point with cluster centroids -0.5 and 1, respectively. But there is of course the "symmetric" fixed point with clusters $\{-1\}$ and $\{0, 1\}$ that has cluster centroids -1 and 0.5, respectively. The exact symmetry here is not necessary. Even if we moved the point 0 slightly the problem would persist.
Many people came up with this or similar solutions. We deducted points if you used more than three points, if it was not clear where the points were located, of if the example was in higher dimensions. Also, a few people gave examples with only two points. This is too simple.