TUM EI 70360: Machine Learning and Optimization
Fall 2023
Lecturer: Reinhard Heckel
Teaching Assistant: Tobit Klug

Problem Set 5
Issued: Tuesday, Nov. 14, 2023
Due: Thursday, Nov. 23, 2023

Problem 1 (Linear separability)**.** Suppose we run the hard-margin SVM (without kernels) on n many d-dimensional data points belonging to two classes. The algorithm finds a solution. After which of the following transformations of the data points is the algorithms still guaranteed to find a solution (not necessarily the same solution)?

(a) Centering the data points by offseting each feature by the corresponding sample average.

(b) Transforming the data to a lower dimensional space by removing a feature.

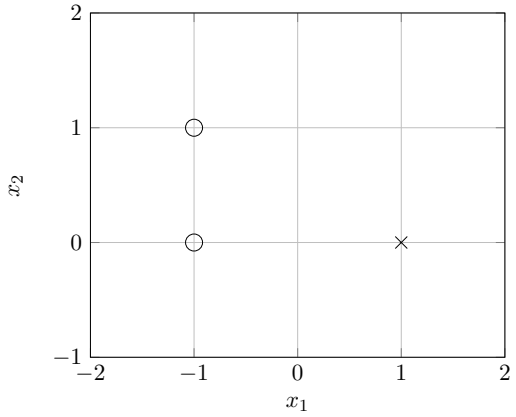
(c) Adding a feature.

(d) Scaling all features with $\alpha > 0$.

Problem 2 (Hard-margin SVM)**.** Suppose we want to solve a binary classification problem using a hard-margin SVM with the linear kernel. The data points are given in the following plot, where the classes are represented by " $\circ$ " and " $\times$ ".

(a) Draw the maximum margin hyperplane and mark it by $\mathcal{H}$.

(b) Suppose the data point $([2,0]^T, \text{" } \times \text{ "})$ is added. Draw the maximum margin hyperplane for the new set of points and mark it by $\mathcal{H}'$.



Problem 3 (Logistic regression)**.**

1. Show that if the (loss) function $L(y, t)$ is convex in t for every y, then the function

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, \langle \mathbf{w}, \mathbf{x}_i \rangle + b) \tag{1}$$

   is convex in $\boldsymbol{\theta} = [b\ \mathbf{w}]^T$.

2. Show that if we choose the loss function as $L(y, t) = \log(1+\exp(-yt))$, then the expression in equation (1) is proportional to the negative log-likelihood for logistic regression. Minimizing the empirical loss associated with a training set as in equation (1) is called empirical loss minimization, or commonly empirical risk minimization (ERM). This problem shows that ERM with the logistic loss is equivalent to logistic regression.

   Note: In the above expression, y is assumed to be -1 or 1. In the logistic regression notes we had $y = 0$ or 1.

Problem 4 (Kernelized perceptrons)**.**

1. Consider a binary classification problem with $y \in \{-1, 1\}$. Suppose that g is a subgradient of a convex loss function $L(y, \langle \boldsymbol{\theta}, \mathbf{x} \rangle)$ with respect to $\boldsymbol{\theta}$. If we have

$$g(\boldsymbol{\theta}) = \begin{cases} -y\mathbf{x} & \langle \boldsymbol{\theta}, \mathbf{x} \rangle\, y < 0 \\ 0 & \text{otherwise} \end{cases}$$

   then what is $L(y, \langle \boldsymbol{\theta}, \mathbf{x} \rangle)$?

   Hint: A vector $g \in \mathbb{R}^d$ is a subgradient of $f : \mathbb{R}^d \to \mathbb{R}$ at $\mathbf{x}$ if for all $\mathbf{z}$,

$$f(\mathbf{z}) \geq f(\mathbf{x}) + g^T(\mathbf{z} - \mathbf{x}).$$

   If f is convex and differentiable, then its gradient at $\mathbf{x}$ is a subgradient. But a subgradient can exist even when f is not differentiable at $\mathbf{x}$, as illustrated in Figure 1.

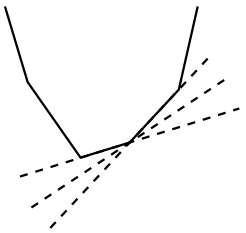


Figure 1: The slope of each dased line is a subgradient.

2. The perceptron uses a hypothesis of the form $h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\langle \boldsymbol{\theta}, \mathbf{x} \rangle)$, where $\text{sign}(t) = 1$ if $t \geq 0$ and -1 otherwise. In this problem we will consider a stochastic subgradient method-like implementation of the perceptron algorithm where each update to the parameters $\boldsymbol{\theta}$ is made using only one training example. However, unlike the stochastic subgradient method, the

perceptron algorithm will only make one pass through the entire training set, which has n observations. The update rule for this version of the perceptron algorithm is given by

$$\boldsymbol{\theta}^{(i+1)} := \begin{cases} \boldsymbol{\theta}^{(i)} + \alpha y^{(i+1)} \mathbf{x}^{(i+1)} & \text{if } h_{\boldsymbol{\theta}^{(i)}}(\mathbf{x}^{(i+1)}) y^{(i+1)} < 0 \\ \boldsymbol{\theta}^{(i)} & \text{otherwise,} \end{cases}$$

where $\boldsymbol{\theta}^{(i)}$ is the value of the parameters after the algorithm has seen the first i training examples. Here, we have that α is the step size and $i \in \{1, 2, \ldots, n\}$. Prior to seeing any training examples, $\boldsymbol{\theta}^{(0)}$ is initialized to 0.

Let k be a positive definite kernel corresponding to some very high-dimensional feature mapping ϕ. Suppose ϕ is so high-dimensional (say, infinite dimensional) that it is infeasible to ever represent $\phi(\mathbf{x})$ explicitly. We would like to apply the "kernel trick" to the perceptron to make it work in the high-dimensional feature space ϕ, but without ever explicitly computing $\phi(\mathbf{x})$. [Note: You do not have to worry about the intercept term. If you like, think of ϕ as having the property that $\phi_0(\mathbf{x}) = 1$ so that this is taken care of.]

How would you (implicitly) represent the high-dimensional parameter vector $\boldsymbol{\theta}^{(i)}$, including how the initial value of $\boldsymbol{\theta}^{(0)} = 0$ is represented? (note that $\boldsymbol{\theta}^{(i)}$ is now a vector whose dimension is the same as the feature vectors $\phi(\mathbf{x}^{(i)})$)

3. How would you efficiently make a prediction on a new input $\mathbf{x}^{(i+1)}$? In other words, how would you compute $h_{\boldsymbol{\theta}^{(i)}}(\mathbf{x}^{(i+1)}) = \text{sign}\left(\left\langle \boldsymbol{\theta}^{(i)}, \phi(\mathbf{x}^{(i+1)}) \right\rangle\right)$, using your representation of $\boldsymbol{\theta}^{(i)}$?

4. How would you modify the update rule given above to perform an update to $\boldsymbol{\theta}$ on a new training sample $(\mathbf{x}^{(i+1)}, y^{(i+1)})$? In other words, how do you compute

$$\boldsymbol{\theta}^{(i+1)} := \begin{cases} \boldsymbol{\theta}^{(i)} + \alpha y^{(i+1)} \phi(\mathbf{x}^{(i+1)}) & \text{if } h_{\boldsymbol{\theta}^{(i)}}(\phi(\mathbf{x}^{(i+1)})) y^{(i+1)} < 0 \\ \boldsymbol{\theta}^{(i)} & \text{otherwise,} \end{cases}$$

feasibly?