HOMEWORK SET 2

CSCI 5525 Advanced Machine Learning (Spring 2021)

Due 11:59 pm, Mar 19 2021

Instruction Typesetting your homework in LaTeX is optional but encouraged, and you need to submit it as a single PDF file in Canvas. For programming, include all your codes and running results in a single Jupyter notebook file and submit it alongside the main PDF (since Jupyter notebook also allows text editing, feel free to put your textual answers inside the Jupyter notebook sometimes). No late submission will be accepted.

For each problem, your should acknowledge your collaborators if any. For problems containing multiple subproblems, there are often close logic connections between the subproblems. So always remember to build on previous ones, rather than work from scratch.

Reminder about notations We will use small letters (e.g., u) for scalars, small boldface letters (e.g., $\boldsymbol{a}$) for vectors, and capital boldface letters (e.g., $\boldsymbol{A}$) for matrices. For a matrix $\boldsymbol{A}$, $\boldsymbol{a}^i$ (supscripting) means its i-th row as a *row vector*, and $\boldsymbol{a}_j$ (subscripting) means the j-the column as a column vector, and a_{ij} means its (i,j)-th element. $\mathbb{R}$ is the set of real numbers. $\mathbb{R}^n$ is the space of n-dimensional real vectors, and similarly $\mathbb{R}^{m\times n}$ is the space of $m \times n$ real matrices. The dotted equal sign $\doteq$ means defining.

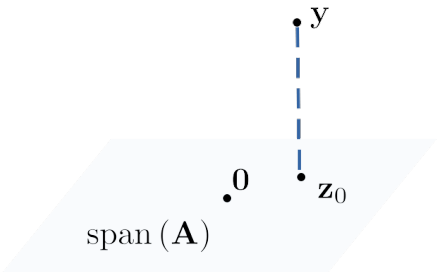
Problem 1 (Least squares problem; 7/12) Consider the least-squares problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \; f(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \boldsymbol{A} \in \mathbb{R}^{m\times n}. \tag{1}$$

In class, we showed that any local (also global, as f is convex) minimizer $\boldsymbol{x}_0$ of f must obey the first-order necessary condition $\nabla f(\boldsymbol{x}_0) = 2\boldsymbol{A}^\intercal(\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{y}) = \boldsymbol{0}$, or $\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{A}^\intercal\boldsymbol{y}$.

(a) Assume $m \geq n$. Show that $\boldsymbol{A}$ has column full rank, i.e., with linearly independent columns, if and only if $\boldsymbol{A}^\intercal\boldsymbol{A}$ is invertible. (Hint: consider using compact SVD, or the fact that if a square matrix $\boldsymbol{M}$ is invertible if and only if the only solution of $\boldsymbol{M}\boldsymbol{z} = 0$ is $\boldsymbol{z} = \boldsymbol{0}$, i.e., if and only if $\boldsymbol{M}$ has a trivial null space. (1/12)

(b) Assume $m \geq n$ and $\boldsymbol{A}$ has column full rank. Then columns of $\boldsymbol{A}$ span an n-dimensional subspace of $\mathbb{R}^m$.



Since any point on the subspace can be written in the form of $\boldsymbol{A}\boldsymbol{x}$ for a certain $\boldsymbol{x}$, geometrically the optimization problem in Eq. (1) is to compute the distance of a given point $\boldsymbol{y} \in \mathbb{R}^m$ to the subspace $\mathrm{span}(\boldsymbol{A})$, i.e.,

$$\min_{\boldsymbol{z}\in\mathrm{span}(\boldsymbol{A})} \; g(\boldsymbol{z}) = \|\boldsymbol{y} - \boldsymbol{z}\|_2^2. \tag{2}$$

Let $\boldsymbol{z}_0$ be a minimizer for Eq. (2). Can we provide an analytic form for $\boldsymbol{z}_0$? Is it unique and why or why not? In $\mathbb{R}^m$, two vectors $\boldsymbol{u}, \boldsymbol{v}$ are orthogonal to each other if $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = 0$. Prove that $\langle \boldsymbol{z}_0 - \boldsymbol{y}, \boldsymbol{w}\rangle$ for any $\boldsymbol{w} \in \mathrm{span}(\boldsymbol{A})$, i.e., the vector $\boldsymbol{z}_0 - \boldsymbol{y}$ is orthogonal to the space $\mathrm{span}(\boldsymbol{A})$—here, $\boldsymbol{z}_0$ is called the orthogonal projection of $\boldsymbol{y}$ onto the subspace. (1/12)

(c) Fix a random seed, and set $m = 100, n = 50$. Generate $\boldsymbol{A}$ and $\boldsymbol{y}$ as iid standard normal. Implement gradient descent, with backtracking line search as the step size rule. Please submit your code, and plot the objective value vs. iterate. (1/12)

(d) When $m < n$, there is no unique global minimizer for f. Assume $\boldsymbol{x}_0$ is a global minimizer. Provide write down the set of all global minimizers and also prove the set indeed contains all global minimizers. (Hint: when constructing the set, start from the optimality condition and derive what condition the difference of two global minimizers must satisfy.) (1/12)

(e) Assume $m < n$ and we run gradient descent to optimize $f(\boldsymbol{x})$ starting with $\boldsymbol{x}^{(0)} = \boldsymbol{0}$ and with a fixed step size t.

(i) Write down the analytic forms of $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$, and $\boldsymbol{x}^{(3)}$ and prove that $\boldsymbol{x}^{(k)} \in \mathrm{row}(\boldsymbol{A})$ for all $k \geq 0$, i.e., all iterates will stay in $\mathrm{row}(\boldsymbol{A})$. (1/12)

(ii) When t is sufficiently small, or backtracking line search is implemented, the sequence $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots$ will converge to a local (and also global) minimizer of $f(\boldsymbol{w})$. Obviously, the limit $\boldsymbol{x}_*$ also lies in $\mathrm{row}(\boldsymbol{A})$. Prove that $\boldsymbol{x}_*$ is global minimizer of $f(\boldsymbol{x})$ with the smallest ℓ_2 norm among all global minimizers. (Hint: any $\boldsymbol{x} \in \mathbb{R}^n$ can be decomposed as two orthogonal components: $\boldsymbol{x} = \boldsymbol{x}_r + \boldsymbol{x}_n$, where $\boldsymbol{x}_r \in \mathrm{row}(\boldsymbol{A})$ and $\boldsymbol{x}_n \in \mathrm{null}(\boldsymbol{A})$.) (1/12)

Why is this interesting? Although in our setting $f(\boldsymbol{x})$ does not have a unique minimizer, a specific algorithm—gradient descent, with specific initialization—zero (in fact, the above results hold for any initialization lying the $\mathrm{row}(\boldsymbol{A})$) and appropriate step sizes, returns a solution with a special property—least ℓ_2 norm, which is simple in some sense. This phenomenon of algorithmic biases toward certain "simple" solutions on problems with many solutions is called *implicit regularization*. This is the basis for one of many emerging theories explaining why deep neural networks generalize well despite that the network capacities well exceed the intrinsic data complexities.

(f) Assume $m < n$ again and so global minimizers for f abound. In statistics, this kind of problems where the number of data points is (far) less than the input (feature) dimension is called *high-dimensional problems* (other fields may use the term in different ways). Reasonable regularization is always needed for high-dimensional problems to bias the estimation procedure toward solutions with certain desired properties. For our least-squares problem Eq. (1), one possibility is seeking (approximately) sparse solutions by adding in ℓ_1 norm regularization:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \frac{1}{m} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 . \tag{3}$$

This is the famous *Lasso* problem. Suppose features (i.e., columns of $\boldsymbol{A}$) are gene expressions of patients and the output is the probability of getting liver cancer. If the solution $\boldsymbol{x}_*$ we find is indeed sparse, i.e., containing very few large entries and the rest entries negligibly small. Then only the genes/columns corresponding to the large entries of $\boldsymbol{x}_*$ are important for predicting the cancer probability. So Lasso and its variants are often used for feature selection.

Fix a random seed, and set $m = 30, n = 100$. Generate $\boldsymbol{A}$ as iid standard normal, a groundtruth $\boldsymbol{x}_0$ as iid Bernoulli with rate 0.2 (i.e., assuming 1 with probability 0.2, and otherwise assuming 0), and so $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ is iid normal with mean 0 and variance 0.5, i.e., $\mathsf{N}(0, 0.5)$. Implement Lasso with the `CVXPY` package, with reference to this example https://www.cvxpy.org/examples/machine_learning/lasso_regression.html. Try 3 different values—ideally

with different orders of magnitude—for λ, and plot the groundtruth $\boldsymbol{x}_*$, and the $\boldsymbol{x}$'s you obtain *in the same stem plot*; check out this link https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.stem.html for making stem plots in Python. What do you observe with increasing λ? (1/12)

Problem 2 (Logistic regression; 5/12)

(a) Recall the maximum likelihood estimation (MLE) principle: given observation data, maximize the probability density (or mass, depending on if the distribution is continuous or discrete) function $p\left(\text{observed data}\right)$—it is the *likelihood function* of unknown parameters $\boldsymbol{\theta}$; we often make the dependency on $\boldsymbol{\theta}$ explicit by writing $p\left(\text{observed data}; \boldsymbol{\theta}\right)$.

For a supervised learning problem, let's assume the inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ are sampled iid from a fixed but unknown distribution $\mathcal{D}_{\boldsymbol{x}}$. Then, each $\boldsymbol{y}_i$ for $i \in [N]$ is sampled from the conditional distribution $\mathcal{D}_{\boldsymbol{y}}\left(\boldsymbol{x}_i, \boldsymbol{w}\right)$, which depends on both $\boldsymbol{x}_i$ and a set of parameters $\boldsymbol{w}$. Show that

$$\max_{\boldsymbol{w}} \; p\left(\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N ; \boldsymbol{w}\right) \tag{4}$$

is equivalent to

$$\max_{\boldsymbol{w}} \; \prod_{i=1}^N p\left(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \boldsymbol{w}\right). \tag{5}$$

This is sometimes called *maxmium conditional likelihood* (MCL) principle. (Hint: note that $p\left(\boldsymbol{x}, \boldsymbol{y}\right) = p\left(\boldsymbol{x}\right) p\left(\boldsymbol{y} \mid \boldsymbol{x}\right)$. 1/12)

(b) Now let's apply MLE to derive logistic regression. Since the logistic function $\phi(z) = \frac{1}{1+e^{-z}}$ takes value in $[0, 1]$, we can use it to naturally model probability. Assume our $\{\boldsymbol{x}_i\}_{i=1}^N$ are sampled from $\mathcal{D}_{\boldsymbol{x}}$, and y_i takes value in $\{1, -1\}$ for all $i \in [N]$ and obeys

$$p\left(y = 1 \mid \boldsymbol{x}\right) = \phi \circ \langle \boldsymbol{w}, \boldsymbol{x} \rangle = \frac{1}{1 + \exp\left(-\langle \boldsymbol{w}, \boldsymbol{x} \rangle\right)}. \tag{6}$$

Obviously, $p\left(y = -1 \mid \boldsymbol{x}\right) = 1 - p\left(y = 1 \mid \boldsymbol{x}\right)$. Show that applying MCL above yields the logistic regression problem we introduced in class:

$$\min_{\boldsymbol{w}} \; \sum_{i=1}^N \log\left(1 + \exp\left(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle\right)\right). \tag{7}$$

(Hint: think of how to derive a unified form of $p\left(y \mid \boldsymbol{x}\right)$ for convenient analysis; 1/12)

(c) Assume we solve problem (7) and get $\boldsymbol{w}_0$. We shall use this for classification on test data. For any data point $\boldsymbol{x}$, a natural classification scheme is:

$$y = \begin{cases} 1 & p\left(y = 1 \mid \boldsymbol{x}\right) > 0.5 \\ -1 & \text{otherwise} \end{cases}. \tag{8}$$

Show that this classifier is linear, i.e., taking the form $\boldsymbol{x} \mapsto \text{sign}\left(\langle \boldsymbol{w}_0, \boldsymbol{x} \rangle\right)$. (1/12)

(d) Derive the gradient of problem (7), and implement gradient with backtracking linear search step size—this will be used in the next subproblem. (1/12)

(e) Load the breast cancer classification dataset in scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html#sklearn.datasets.load_breast_cancer. Test and report the classification accuracy of your logistic regression implementation, and compare it with that by the built-in logistic regression module https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Don't panic if you see slightly different results, as scikit-learn implements a regularized version; see https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression. (1/12)