HOMEWORK SET 1

CSCI5525 Machine Learning: Analysis and Methods (Fall 2024)

Due 11:59 pm, Oct 06 2024

Instruction Your writeup, either typeset or scanned, should be a single PDF file. For problems requiring coding, organize all codes for each problem into a separate Jupyter notebook file (i.e., `.ipynb` file). Your submission to Gradescope should include the single PDF and all notebook files—**please DO NOT zip them**! No late submission will be accepted. For each problem, you should acknowledge your collaborators—**including AI tools**, if any.

About the use of AI tools You are strongly encouraged to use AI tools—they are becoming our workspace friends, such as ChatGPT (`https://chat.openai.com/`), Claude (`https://claude.ai/chats`), and Github Copilot (`https://github.com/features/copilot`), to help you when trying to solve problems. It takes a bit of practice to ask the right and effective questions/prompts to these tools; we highly recommend that you go through this popular free short course **ChatGPT Prompt Engineering for Developers** offered by `https://learn.deeplearning.ai/` to get started.

If you use any AI tools for any of the problems, you should include screenshots of your prompting questions and their answers in your writeup. The answers provided by such AI tools often contain factual errors and reasoning gaps. **So, if you only submit an AI answer with such bugs for any problem, you will obtain a zero score for that problem**. You obtain the scores only when you explain the bugs and also correct them in your own writing. You can also choose not to use any of these AI tools, in which case we will grade based on the efforts you have made.

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.

Recommended references for linear algebra The book series on linear algebra by Prof. Gilbert Strang (MIT) is highly recommended: they give you the right intuitions and insights about linear algebra, especially after you take the first introductory course to linear algebra.

- **Introduction to Linear Algebra** `https://math.mit.edu/~gs/linearalgebra/ila6/indexila6.html`
- **Linear Algebra for Everyone** `https://math.mit.edu/~gs/everyone/`
- **Linear Algebra and Learning from Data** `https://math.mit.edu/~gs/learningfromdata/`

You don't have to buy these books; you can find a full collection of related video lectures that are equally useful from Prof. Strang's website `https://math.mit.edu/~gs/`.

Problem 1 (Matrix norms, inner products, traces; $4/15$) Recall that for any vector $\boldsymbol{v} \in \mathbb{R}^n$ and any $p \geq 1$, the ℓ_p norm of $\boldsymbol{v}$ is defined as $\|\boldsymbol{v}\|_p \doteq \left(\sum_i |v_i|^p\right)^{1/p}$. The cases where $p = 1, 2, \infty$ are often used. When $p = 2$, it is also called the Euclidean norm. Similar norms can be defined for matrices. Particularly, the direct generalization of the vector Euclidean norm is the *Frobenius norm* defined as

$$\|\boldsymbol{M}\|_F \doteq \sqrt{\sum_{ij} m_{ij}^2}$$

for a matrix $\boldsymbol{M}$. On the other hand, the inner product of matrices is defined similarly to that of vectors. For $\boldsymbol{A}, \boldsymbol{B}$ of the same size, $\langle \boldsymbol{A}, \boldsymbol{B} \rangle \doteq \sum_{ij} a_{ij} b_{ij}$. Obviously, $\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \langle \boldsymbol{B}, \boldsymbol{A} \rangle$ and $\|\boldsymbol{M}\|_F = \sqrt{\langle \boldsymbol{M}, \boldsymbol{M} \rangle}$. A third notion of interest is the matrix trace, $\operatorname{tr}(\boldsymbol{M}) = \sum_i m_{ii}$, i.e., sum of the diagonal entries, which is only defined for square matrices.

Please provide detailed steps with justification for all subproblems below; jumping to the final results leads to a zero score. Also, if we ask you to use certain facts to prove a thing, you have to use these facts (perhaps plus others); otherwise you get a zero score.

(a) Show that $\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \operatorname{tr}(\boldsymbol{A}^\intercal \boldsymbol{B})$ and that $\|\boldsymbol{M}\|_F = \sqrt{\operatorname{tr}(\boldsymbol{M}^\intercal \boldsymbol{M})}$. $(1/15)$

(b) Using (a) and perhaps additional facts to show that $\operatorname{tr}(\boldsymbol{A}^\intercal \boldsymbol{B}) = \operatorname{tr}(\boldsymbol{B}^\intercal \boldsymbol{A})$. $(0.5/15)$

(c) Assume $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same size. In general, $\boldsymbol{A}\boldsymbol{B}^\intercal$ and $\boldsymbol{B}^\intercal \boldsymbol{A}$ have different sizes, but $\operatorname{tr}(\boldsymbol{A}\boldsymbol{B}^\intercal) = \operatorname{tr}(\boldsymbol{B}^\intercal \boldsymbol{A})$. Show it using (a) and perhaps additional facts! $(0.5/15)$

(d) Using (c) and perhaps additional facts to show that $\operatorname{tr}(\boldsymbol{M}_1\boldsymbol{M}_2\boldsymbol{M}_3) = \operatorname{tr}(\boldsymbol{M}_3\boldsymbol{M}_1\boldsymbol{M}_2) = \operatorname{tr}(\boldsymbol{M}_2\boldsymbol{M}_3\boldsymbol{M}_1)$, assuming that the sizes of $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and $\boldsymbol{M}_3$ are compatible with all the matrix multiplications. This is known as the *cyclic property* of matrix traces. $(0.5/12)$

(e) For any matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ of compatible sizes, we always have $\langle \boldsymbol{A}\boldsymbol{C}\boldsymbol{B}, \boldsymbol{D} \rangle = \langle \boldsymbol{C}\boldsymbol{B}, \boldsymbol{A}^\intercal \boldsymbol{D} \rangle = \langle \boldsymbol{A}\boldsymbol{C}, \boldsymbol{D}\boldsymbol{B}^\intercal \rangle$, i.e., we can always move the **leading** matrix of one side of the inner product to the other side as **leading** matrix **once transposed**, and similarly the **trailing** matrix to the other side as **trailing** matrix once **transposed**. Using (d) and perhaps additional facts to show it. $(0.5/15)$

(f) For $\boldsymbol{M}$, let's perform a *compact SVD* (if not sure, check Wikipedia! `https://en.wikipedia.org/wiki/Singular_value_decomposition#Compact_SVD`) to obtain $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal$, so that $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal (not necessarily square) matrices, i.e., $\boldsymbol{U}^\intercal \boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{V}^\intercal \boldsymbol{V} = \boldsymbol{I}$. Use the cyclic property of trace and that $\|\boldsymbol{M}\|_F = \sqrt{\operatorname{tr}(\boldsymbol{M}^\intercal \boldsymbol{M})}$ to show that

$$\|\boldsymbol{M}\|_F = \sqrt{\sum_{i=1}^{r} \sigma_i^2},$$

assuming the rank of $\boldsymbol{M}$ is r. Here, σ_i's are the singular values of $\boldsymbol{M}$. $(1/15)$

Problem 2 (Computation of Jacobian, gradient, and Hessian; 4/15) To derive Jacobian, gradient, and Hessian, you are free to deploy any techniques or their mixtures that we describe in the class. However, the perturbation-expansion (Taylor-expansion) technique could be (much) more efficient for some cases.

Background on convexity: A twice-differentiable function $f(\boldsymbol{x})$ is convex if $\nabla^2 f(\boldsymbol{x})$ is positive semidefinite (i.e., $\nabla^2 f(\boldsymbol{x}) \succeq \boldsymbol{0}$) for all $\boldsymbol{x}$. If $\nabla f^2(\boldsymbol{x})$ is positive definite (i.e., $\nabla^2 f(\boldsymbol{x}) \succ \boldsymbol{0}$) for all $\boldsymbol{x}$, then f is said to be strongly convex and f has a unique minimizer.

(a) Let $\boldsymbol{A}$ be a square, but not necessarily symmetric, matrix. Deriving the gradient and Hessian of the quadratic function $f(\boldsymbol{x}) = \boldsymbol{x}^\intercal \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^\intercal \boldsymbol{x}$. Please include your calculation details. (Hint: note that Hessian must be a symmetric matrix.) $(1/15)$

(b) Let $p(\boldsymbol{x}; \boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}^\intercal \boldsymbol{x}}}{1+e^{\boldsymbol{\beta}^\intercal \boldsymbol{x}}}$. The log-likelihood function in logistic regression is (assuming N training points $\{(\boldsymbol{x}_i, y_i)\}_{i=1,\dots,N}$)

$$f(\boldsymbol{\beta}) = \sum_{i=1}^{N} \left[y_i \log p(\boldsymbol{x}_i; \boldsymbol{\beta}) + (1-y_i) \log (1 - p(\boldsymbol{x}_i; \boldsymbol{\beta})) \right] \tag{1}$$

$$= \sum_{i=1}^{N} \left[y_i \boldsymbol{\beta}^\intercal \boldsymbol{x}_i - \log\left(1 + e^{\boldsymbol{\beta}^\intercal \boldsymbol{x}_i}\right) \right]. \tag{2}$$

Derive the gradient and Hessian of $f(\boldsymbol{\beta})$. Please include your calculation details. $(1/15)$

For logistic regression, we are going to maximize $f(\boldsymbol{\beta})$, which is equivalent to minimize $-f(\boldsymbol{\beta})$. Does the minimization problem has a unique minimizer or not? $(0.5/15)$

(c) Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with $m < n$. Then given any $\boldsymbol{y} \in \mathbb{R}^m$, the least-squares problem $\min_{\boldsymbol{x}} \ \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$ has infinitely many solutions, as $\boldsymbol{A}$ has a nontrivial null space. Now let's say we want a solution with a small ℓ_2 norm, then it is reasonable to put a regularization/penalty on the ℓ_2 norm of $\boldsymbol{x}$, leading to

$$\min_{\boldsymbol{x}} \ \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_2^2 \tag{3}$$

with a chosen $\lambda > 0$. This is the notable *ridge regression*. Now we know that for an unconstrained first-order differentiable function $g(\boldsymbol{x})$, any of its local minimizer $\boldsymbol{x}_*$ must satisfy the *first-order optimality condition*: $\nabla g(\boldsymbol{x}_*) = \mathbf{0}$. Use this to derive a candidate local minimizer $\boldsymbol{x}_*$ $(0.5/15)$. Is it a local minimizer? A global minimizer? Is it unique? Why? $(1/15)$

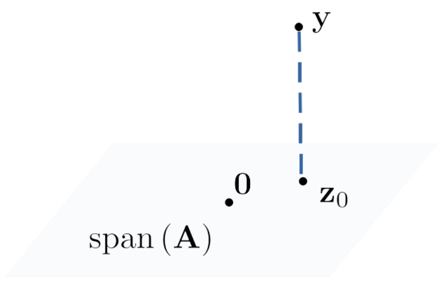
Problem 3 (Least squares problem; $5.5/15$) Consider the least-squares problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \ f(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \boldsymbol{A} \in \mathbb{R}^{m \times n}. \tag{4}$$

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 $\nabla f(\boldsymbol{x}_0) = 2\boldsymbol{A}^\intercal(\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{y}) = \mathbf{0}$, or $\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{A}^\intercal\boldsymbol{y}$.

(a) Assume $m \ge n$. 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 $\boldsymbol{z} = \mathbf{0}$, i.e., if and only if $\boldsymbol{M}$ has a trivial null space. $(1/15)$

(b) Assume $m \ge n$ and that $\boldsymbol{A}$ 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. (4) is to compute the (squared) ℓ_2 distance of a given point $\boldsymbol{y} \in \mathbb{R}^m$ to the subspace $\operatorname{span}(\boldsymbol{A})$, i.e.,

$$\min_{\boldsymbol{z} \in \operatorname{span}(\boldsymbol{A})} \ g(\boldsymbol{z}) = \|\boldsymbol{y} - \boldsymbol{z}\|_2^2. \tag{5}$$

Let $\boldsymbol{z}_0$ be a minimizer for Eq. (5). Can we provide a closed 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 $\langle \boldsymbol{z}_0 - \boldsymbol{y}, \boldsymbol{w} \rangle = 0$ for any $\boldsymbol{w} \in \operatorname{span}(\boldsymbol{A})$, i.e., the vector $\boldsymbol{z}_0 - \boldsymbol{y}$ is orthogonal to the space $\operatorname{span}(\boldsymbol{A})$—here, $\boldsymbol{z}_0$ is called the orthogonal projection of $\boldsymbol{y}$ onto the subspace. $(1/15)$

(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. $(0.5/15)$

(d) When $m < n$, there is no unique global minimizer for f. Assume $\boldsymbol{x}_0$ is a global minimizer. Please 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.) (1/15)

(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 $\mathrm{row}(\boldsymbol{A})$. (0.5/15)

(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, which is 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 $\boldsymbol{x}_n \in \mathrm{null}(\boldsymbol{A})$.) (1/15)

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 in $\mathrm{row}(\boldsymbol{A})$) 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 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 problem 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. (4), one possibility is seeking (approximately) sparse solutions by adding an ℓ_1 norm regularization:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \frac{1}{m} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 . \tag{6}$$

This is the famous *Lasso* problem. Suppose that 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 (find the installation instruction here: `https://www.cvxpy.org/install/index.html`), 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 $\boldsymbol{x}_0$, 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? $(0.5/15)$

Problem 4 (Robust linear regression; $1.5/15$) Given data points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$. Linear regression tries to find a linear function of $\boldsymbol{x}$, i.e., $\boldsymbol{w}^\intercal \boldsymbol{x}$, so that the collective approximation error $\sum_{i=1}^N \ell\left(\boldsymbol{w}^\intercal \boldsymbol{x}_i, y_i\right)$ is minimized. Geometrically, this fits a linear subspace to the point cloud $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ so that the cumulative error in the y direction can be minimized, as illustrated in Fig. 1. If the data points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ do follow a linear relationship with deviations of comparable magnitude, e.g., for an underlying $\boldsymbol{w}_*$

$$y_i = \boldsymbol{w}_*^\intercal \boldsymbol{x}_i + \varepsilon_i \;\forall\; i = 1, \dots, N$$

with ε_i iid Gaussian with a small variance, the typical least-squares formulation, e.g.,

$$\min_{\boldsymbol{w}} \; \frac{1}{N} \sum_{i=1}^N \left(\boldsymbol{w}^\intercal \boldsymbol{x}_i - y_i\right)^2 \tag{7}$$

is reasonable. But if ε_i's are very different in magnitude across the i's, e.g., coming from heavy-tailed distributions, or due to irregular measurement corruption, $(\cdot)^2$ as loss function may not be appropriate. This is because an extremely large ε_i can dominate the total loss and ruin the estimation of $\boldsymbol{w}_*$.

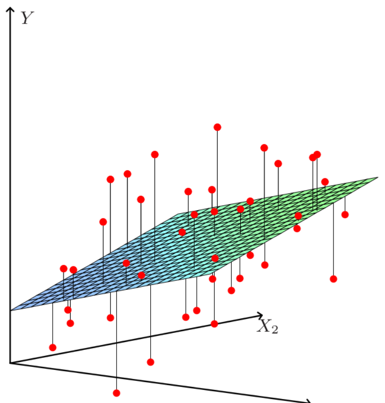


Figure 1: Illustration of linear regression. Figure reproduced from Figure 3.1 of [**?**].

To cure this, an alternative is to use the absolute value, instead of the squares:

$$\min_{\boldsymbol{w}} \; \frac{1}{N} \sum_{i=1}^N \left|\boldsymbol{w}^\intercal \boldsymbol{x}_i - y_i\right|. \tag{8}$$

Compared to Eq. (7), the influence of terms with potential large errors is suppressed and with small errors amplified, so the estimation procedure tends to be more stable. This is often called *least absolute deviations* (LAD) estimation. For more information on this, check out here https://en.wikipedia.org/wiki/Least_absolute_deviations.

Now let's explore the benefit of LAD. *For the sake of reproducibility, please fix a random seed before you start to generate any data.*

(a) Let's first generate an iid normal (i.e., $\mathcal{N}(0,1)$) vector $\boldsymbol{w}_* \in \mathbb{R}^{20}$ and 100 iid normal vectors $\boldsymbol{x}_i \in \mathbb{R}^{20}$. Now produce $y_i = \boldsymbol{w}_*^\intercal \boldsymbol{x}_i + \varepsilon_i$, where ε_i's are iid $\mathrm{Laplace}(0,1)$. $(0.5/15)$

(b) Read this CVXPY example on solving least squares https://www.cvxpy.org/examples/basic/least_squares.html and adapt it for solving LAD with the data generated in (a). To solve LAD, you only need to change the line

```
cost = cp.sum_squares(A @ x - b)
```

to

```
cost = cp.norm1(A @ x - b).
```

For the $\widehat{\boldsymbol{w}}$ estimated from both least-squares and LAD, compute $\left\|\widehat{\boldsymbol{w}} - \boldsymbol{w}_*\right\|_2$, i.e., estimation error for $\boldsymbol{w}_*$. Which method leads to a smaller estimation error? $(1/15)$