HOMEWORK SET 1

CSCI 5525 Advanced Machine Learning (Spring 2021)

Due 11:59 pm, Feb 10 2021

Instruction Typesetting your homework in LaTeX is optional but encouraged, and you need to submit it as a single PDF file in Canvas. 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 (Matrix norms, inner products, traces; 6/12) Recall that for any vector $\boldsymbol{v} \in \mathbb{R}^n$, 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 when $p = 1, 2, \infty$ 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, $\mathrm{tr}\,(\boldsymbol{M}) = \sum_i m_{ii}$, i.e., sum of the diagonal entries, which is only defined for square matrices.

(a) Show that $\langle \boldsymbol{A}, \boldsymbol{B}\rangle = \mathrm{tr}\,(\boldsymbol{A}^\intercal \boldsymbol{B})$ and so $\|\boldsymbol{M}\|_F = \sqrt{\mathrm{tr}\,(\boldsymbol{M}^\intercal \boldsymbol{M})}$. (1/12)

(b) Show that $\mathrm{tr}\,(\boldsymbol{A}^\intercal \boldsymbol{B}) = \mathrm{tr}\,(\boldsymbol{B}^\intercal \boldsymbol{A})$. (1/12)

(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 $\mathrm{tr}\,(\boldsymbol{A}\boldsymbol{B}^\intercal) = \mathrm{tr}\,(\boldsymbol{B}^\intercal \boldsymbol{A})$. Show it! (1/12)

(d) Show that $\mathrm{tr}\,(\boldsymbol{M}_1\boldsymbol{M}_2\boldsymbol{M}_3) = \mathrm{tr}\,(\boldsymbol{M}_3\boldsymbol{M}_1\boldsymbol{M}_2) = \mathrm{tr}\,(\boldsymbol{M}_2\boldsymbol{M}_3\boldsymbol{M}_1)$, 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. (Hint: think of (c)) (1/12)

(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** (if these matrices are complex-valued, should be conjugate transposed), and similarly the **trailing** matrix to the other side as **trailing** matrix once **transposed**. Why? (Hint: think of the above results and also try to remember this important property that will be useful for calculation later) (1/12)

(f) For $\boldsymbol{M}$, let's perform a *compact SVD* (if not sure, check up 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 $\boldsymbol{M}$. (1/12)

Problem 2 (Computation of Jacobian, gradient, and Hessian; 4/12) For all the subproblems, 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 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 for all $\boldsymbol{x}$. If $\nabla f^2(\boldsymbol{x})$ is positive definite 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 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.) (1/12)

(b) Let $p(\boldsymbol{x};\boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}^\intercal\boldsymbol{x}}}{1+e^{\boldsymbol{\beta}^\intercal\boldsymbol{x}}}$. The log-likelihood for logistic regression with two classes is (assuming N samples of the form $(\boldsymbol{x}_i, y_i)$)

$$\begin{aligned} f(\boldsymbol{\beta}) &= \sum_{i=1}^{N} \left[y_i \log p(\boldsymbol{x}_i;\boldsymbol{\beta}) + (1-y_i)\log\left(1 - p(\boldsymbol{x}_i;\boldsymbol{\beta})\right)\right] \\ &= \sum_{i=1}^{N} \left[y_i \boldsymbol{\beta}^\intercal\boldsymbol{x}_i - \log\left(1 + e^{\boldsymbol{\beta}^\intercal\boldsymbol{x}_i}\right)\right]. \end{aligned}$$

Derive the gradient and Hessian of $f(\boldsymbol{\beta})$. Please include your calculation details. (1/12) 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? (0.5/12)

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

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

with a chosen $\lambda > 0$. This is *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*: $\nabla g(\boldsymbol{x}_*) = \boldsymbol{0}$. Use this to derive $\boldsymbol{x}_*$ (1/12). Is the $\boldsymbol{x}_*$ unique? Why? (0.5/12)

Problem 3 (Robust linear regression; 2/12) 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 $\sum_{i=1}^N \ell(\boldsymbol{w}^\intercal\boldsymbol{x}_i, y_i)$ 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.

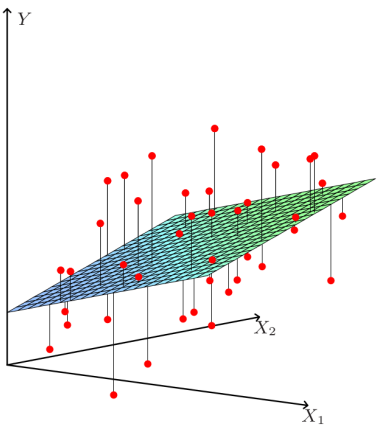

Figure 1: Illustration of linear regression. Figure reproduced from Figure 3.1 of [HTF09].

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 $\boldsymbol{w}_*$

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

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

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

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, the $(\cdot)^2$ as loss function may not be appropriate. This is because one extremely large ε_i may rule the total loss and ruin the estimation of $\boldsymbol{w}_*$.

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

$$\min_{\boldsymbol{w}} \sum_{i=1}^N |\boldsymbol{w}^\intercal \boldsymbol{x}_i - y_i| \,. \tag{2}$$

Compared to Eq. (1), 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 50 iid normal vectors $\boldsymbol{x}_i \in \mathbb{R}^{20}$. Now produce

$$y_i = \boldsymbol{w}_*^\intercal \boldsymbol{x}_i + \varepsilon_i, \tag{3}$$

where ε_i's are iid Laplace $(0, 2)$. (0.5/12)

(b) `CVXPY` is an excellent Python-based modeling framework for solving small-scale convex optimization problems. Follow the instruction here https://www.cvxpy.org/install/index.html and install it in your workspace. (0.5/12)

(c) Read this `CVXPY` example on solving least squares https://www.cvxpy.org/examples/basic/least_squares.html and adapt it for solving our least-squares with the data of (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 $\|\widehat{\boldsymbol{w}} - \boldsymbol{w}_*\|_2$, i.e., estimate error for $\boldsymbol{w}_*$. Which method leads to smaller estimation error? (1/12)

References

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The elements of statistical learning: Data mining, inference, and prediction, second edition*, SPRINGER NATURE, 2009.