Homework 3

Stats 315A, Winter 2026

February 16, 2026

Due date: Monday, February 23rd at 11:59pm on Canvas.

Submission instructions: Upload a `.pdf` file with your answers, and a separate text file or zip file with your code from questions 1 and 4.

Question 1 (Protein Binding Energy Prediction, 30pt)**:**

An important task in computational protein modeling is to predict how mutations to proteins impact the strength of their interactions with other proteins. In this assignment you will fit predictors of binding energies, evaluate them, and submit predictions as part of a Kaggle contest.

Background knowledge related to the task is *not required* to receive full credit for the problem. But for part (d.), there will be extra credit for those who perform well in the prediction competition. For this part, you may find it useful learn more about the task, for example by consulting some of the papers cited herein, using LLM tools, etc. We ask only that you do not attempt to find labels for the test predictions and that you explain your approach.

Data. Each training datapoint represesents a protein binding interaction and is associated with a row of covariates in `protein_train_x.csv` and a response in row of `protein_train_y.csv`. The response (`ddG`) in `protein_train_y.csv` is difference in the binding free energy of the interaction for a mutated protein compared to a wildtype (i.e. unmutated) protein. The covariates file `protein_train_x.csv` contains pre-processed data from the SKEMPI dataset (Jankauskaitė et al., 2019) with several additional covariates. Appendix A provides an explanation of these covariates.

a. **Least-squares baseline, 5 points.**

Using `protein_train_x.csv` and `protein_train_y.csv`, fit a least-squares regression model. Ignore the columns `pdb_code`, `skempi_cluster`, and `position`. You should

(i) Describe the design matrix you constructed and how you fit the model. Did you need to do anything to avoid numerical instability?

(ii) Use cross-validation to estimate the out-of-sample mean squared error. Compare this to the variance on of the responses.

b. **Improving the predictor, 10 points.**

Attempt to improve upon your least squares baseline either on the basis of predictive accuracy on the trainig data, or suitability for making predictions on the test covariates. In your response to this question, you should

(i) Describe the alternative method you have tried, your rationale, and how you have run it.

(ii) Compare the error of your method to the least squares predictor by cross validation.

(iii) Evaluate whether the difference in error may be "due to chance". If you use a statistical test, explain your test (what is the null hypothesis?), and your conclusion.

(iv) Describe which feature or features are most important to your predictor.

c. **Evaluation scenarios, 5 points.** Suppose you speak to two scientists, Alice and Bob, who are interested in using your predictor. Alice wants predictions for mutations to a hypothetical protein with PDB identifier XYZ; there are data on XYZ in `protein_train_x.csv` and `protein_train_y.csv`, but not for the mutations of interest. Bob wants predictions for mutations for proteins not associated with PDB identifiers represented in the training data.

Is your analysis equally informative for both scientists? Describe how you might tailor your analyses to Alice's and Bob's interests.

d. **Out of distribution generalization and Kaggle competition, 10 points.**

The goal of this final part is similar to Bob's goal in part c, to make predictions for several mutations for each of several proteins not represented in the training data. Because ranking the sizes of the effects of mutations is a common goal in protein modeling, performance will be measured by the mean across the test proteins of the Spearman rank correlation of predictions with the test responses.

For this part, we additionally provide `protein_test_x.csv` with features corresponding to those in `protein_train_x.csv` for the test cases. This set comprises a mix of examples from the SKEMPI dataset and separately curated examples.

Address the following

(i) What are two or more possible limitations of the predictors from parts a and b for the present goal?

(ii) Pick one of these limitations and fit a new predictor that addresses it. Describe what you have done.

(iii) Submit predictions for your method to the Kaggle challenge. Credit for this part does not depend on the performance of your method.

Extra credit: Try to improve your score on the held-out data. You may submit predictions up to 3 times, and only your best score will be considered. Points will be awarded to submissions in the top 1/3 of participants, with additional points to the highests scorers. Explain of your best submission (if it differs from what you have described already).

Question 2 (Subgradients *descent*? **7pts**)**:**
Let $\|\cdot\|_1$ denote the ℓ_1 norm on $\mathbb{R}^d$, and let e_i be the i-th standard basis vector.

(a) **Subgradient of the ℓ_1 norm.**
Compute the subdifferential $\partial\|x\|_1$. Then evaluate $\partial\|x\|_1$ at $x = e_i$.

(b) **Subgradient descent is not a descent method.**
Consider $f(x) = \|x\|_1$ at $x = e_1$. Show that any vector of the form

$$g = e_1 + \sum_{i=2}^{n} t_i e_i, \qquad \sum_{i=2}^{n} |t_i| > 1,$$

is an ascent direction for f, meaning that

$$f(e_1 - \alpha g) > f(e_1) \quad \text{for all } \alpha > 0.$$

This example shows that the subgradient method is not a descent method even at points where the optimal value is not reached. In fact, if we were to pick a uniformly random $g \in \partial f(e_1)$, for example, then the probability that g is a descent direction is exponentially small in the dimension d (cf. J. Duchi's lecture notes).

(c) **Best-iterate guarantee.**
Assume we perform k iterations of the subgradient algorithm with step sizes $\alpha_i > 0$. Since it is not a descent algorithm, instead of looking at the final value x_k, it makes sense to look at the value x that achieved the lowest value $f(x)$ across all k iterations. Define

$$x_k^{\text{best}} = \arg\min_{1 \le i \le k} f(x_i), \qquad f_k^{\text{best}} = f(x_k^{\text{best}}).$$

Using Theorem 3.2.7 from J. Duchi's lecture notes (proved in class, cf. lecture 9), prove that under the same conditions (M-lipschitz subgradients)

$$f(x_k^{\text{best}}) - f(x^\star) \le \frac{\|x_1 - x^\star\|_2^2 + \sum_{i=1}^k \alpha_i^2 M^2}{2\sum_{i=1}^k \alpha_i}.$$

(*Hint:* in a list of k elements, how does its minimum value compare to any convex combination of the elements of the list?)

As shown in lecture 9, this bound implies that choosing $\alpha_i = \alpha \propto \frac{1}{\sqrt{k}}$ yields

$$f(x_k^{\text{best}}) - f(x^\star) \lesssim \frac{1}{\sqrt{k}}.$$

Question 3 (Computing an update beyond stochastic gradient descent, **6 points**)**:**

The standard stochastic gradient method iteratively makes a linear approximation to a given loss, then minimizes it plus a quadratic regularization:

$$\theta_\alpha := \operatorname*{argmin}_\theta \left\{ \ell(\theta_0) + g^T(\theta - \theta_0) + \frac{1}{2\alpha}\|\theta - \theta_0\|^2 \right\},$$

where $g \in \partial \ell(\theta_0)$, and which has the trivial solution $\theta_\alpha = \theta - \alpha g$. In this problem, we consider instead solving an update that applies when the loss ℓ is nonnegative, so we model it by a *nonnegative* function:

$$\theta_\alpha := \operatorname*{argmin}_\theta \left\{ \left(\ell(\theta_0) + g^T(\theta - \theta_0)\right)_+ + \frac{1}{2\alpha}\|\theta - \theta_0\|^2 \right\}. \tag{1}$$

See the papers Asi and Duchi (2019b,a) for more about such more sophisticated updates.

(a) Let $b > 0$ and $a \in \mathbb{R}$. Give the minimizer in $t \in \mathbb{R}$ of

$$(a - bt)_+ + \frac{\lambda}{2}t^2.$$

(b) Demonstrate that θ_α as defined in (1) satisfies

$$\theta_\alpha = \theta_0 - tg$$

for some $t \ge 0$.

(c) Use your answers to the previous parts to show that (recall $\ell(\theta_0) \geq 0$)

$$\theta_\alpha = \theta_0 - \min\left\{\frac{\ell(\theta_0)}{\|g\|^2}, \alpha\right\} g.$$

(d) Let $H \in \mathbb{R}^{d\times d}$ be a diagonal matrix with positive entries on the diagonal, $\ell : \mathbb{R}^d \to \mathbb{R}_+$ be a nonnegative loss function, and $g \in \partial\ell(\theta_0)$ be an element of the subdifferential of ℓ at θ_0. Let $\alpha > 0$. Give

$$\theta_\alpha := \operatorname*{argmin}_\theta \left\{ \left(\ell(\theta_0) + g^\top(\theta - \theta_0)\right)_+ + \frac{1}{2\alpha}(\theta - \theta_0)^\top H(\theta - \theta_0) \right\}.$$

Question 4 (Implementing truncated gradient methods **12 points**)**:**

In this question, you will implement two versions of truncated stochastic gradient methods from question 3. We wish to solve the stochastic optimization problem

$$\operatorname*{minimize}_\theta \; L(\theta) := \mathbb{E}_P[\ell(\theta, Z)] = \int \ell(\theta, z) dP(z),$$

where Z is a random variable and ℓ is nonnegative (with minimal value $0 = \inf_\theta \ell(\theta, z)$ for all z). For these methods, we iteratively draw $Z_k \sim P$, i.i.d., set $g_k \in \partial\ell(\theta_k, Z_k)$, then update

$$\theta_{k+1} = \operatorname*{argmin}_\theta \left\{ \left(\ell(\theta_k) + \langle g_k, \theta - \theta_k\rangle\right)_+ + \frac{1}{2\alpha_k}(\theta - \theta_k)^\top H_k(\theta - \theta_k) \right\}, \tag{T-UP}$$

where $H_k \succeq 0$ is a diagonal matrix.

i. For the truncated stochastic gradient method, $H_k = I$ is the identity matrix.

ii. For truncated AdaGrad, $H_k = \operatorname{diag}(\sum_{i=1}^k g_i g_i^\top)^{1/2} + \epsilon I$ has jth diagonal entry equal to the square root of the sum of squared jth gradient component including iteration k, where $\epsilon \geq 0$ (typically, $\epsilon \approx 10^{-10}$) avoids `NaN`s.

The file `beyond_gd_starter.py` provides starter code for implementing the update (T-UP) in `PyTorch`, with code you should fill in marked `Your code here` in the two classes `TruncatedSGD` and `TruncatedAdagrad`. These classes subclass the `PyTorch` optimizer class, meaning that they need only implement the `step` method. We provide wrapper code in the file `mnist_experiments.py` to load and fit either a small multi-layer perceptron or a small convolutional neural network on the MNIST digit recognition dataset. This starter code will automatically leverage GPUs if you have them,[1] though this only makes a difference for convolutional networks.

(a) Implement the `step` update for `TruncatedSGD`.

(b) Implement the `step` update for `TruncatedAdagrad`. Be careful about exactly which square roots you take in implementing the updates (T-UP).

(c) Using the method `FitNN` with the default learning rate (i.e., stepsize), run the following 8 experiments: fit a multi-layer perceptron or a convolutional network with the optimization methods Adam, Adagrad, TruncatedSGD, and TruncatedAdagrad. Include print-outs of your results.

[1] If you are running experiments on a Mac, you will need to use the nightly build of `PyTorch` to have access to Apple's `MPS`.

For reference, on a 2021 MacBook Pro, our code fits the MLP in roughly 2–4 seconds per epoch, for a total of approximately 20 seconds; the ConvNet architecture requires approximately 15 seconds per epoch using the GPUs available through `MPS`, for a total of approximately 90 seconds. It is twice as slow using only the CPU. You should achieve roughly 99% test accuracy using the Convolutional network and any of Adam, AdaGrad, and TruncatedAdagrad.

Question 5 (Automatic differentiation, **3 pts**)**:**
Consider the `Value` python class from the automatic differentiation notebook from class (this is posted on Canvas).

(a.) Complete the below implementation of the method `exp` that for a `Value` object `A` returns a new `Value` with data equal to `math.exp(A.data)`.

```
class Value:
    def __init__(self, data, _children=(), _op="", local_ders=()):
        self.data = data
        self._prev = _children
        self._op = _op
        self.local_ders = local_ders

    ...

    def exp(self):
        local_der = # Your line here
        out = Value(math.exp(self.data), _children=(self,), local_ders=(local_der,))
        return out
```

Your answer should be the single line beginning `local_der =` .

Question 6 (Time and collaboration, **2 points**)**:**

(a) How many hours in total did you spend on this assignment? (This will help to calibrate future assignments.)

(b) With whom (if anyone) did you collaborate on this assignment?

References

Hilal Asi and John C. Duchi. The importance of better models in stochastic optimization. *Proceedings of the National Academy of Sciences*, 2019a.

Hilal Asi and John C. Duchi. Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity. *SIAM Journal on Optimization*, 2019b.

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 2022.

Arthur Deng, Karsten D Householder, Fang Wu, K Christopher Garcia, and Brian L Trippe. Predicting mutational effects on protein binding from folding energy. In *International Conference on Machine Learning*. PMLR, 2025.

Justina Jankauskaitė, Brian Jiménez-García, Justas Dapkūnas, Juan Fernández-Recio, and Iain H. Moal. Skempi 2.0: an updated benchmark of changes in protein–protein binding energy, kinetics and thermodynamics upon mutation. *Bioinformatics*, 2019.

Shitong Luo, Yufeng Su, Zuofan Wu, Chenpeng Su, Jian Peng, and Jianzhu Ma. Rotamer density estimator is an unsupervised learner of the effect of mutations on protein-protein interaction. *bioRxiv*, 2023.

A Explanation of features for binding energy prediction challenge

The covariates columns in `protein_train_x.csv` and `protein_test_x.csv` are as follows:

- `pdb`: The Protein Data Bank (PDB) structure identifier of an x-ray crystal structure of protein complex for which the strength of binding interactions has been measured. The 4 character identifier is followed by the labels of the chains involved in one side of the protein binding interaction (binder 1) and then those of the other side (binder 2). For example `1C1Y_A_B` indicates a binding interaction between chains `A` and `B` in PDB entry `1CIY`.
- `cluster`: SKEMPI cluster identifier.
- `mutation_chain`: The identifier of the chain on which the mutation.
- `mutation_pos`: The position the mutation as the index in processed protein structure file in hte SKEMPI dataset
- `wt_aa`: Wildtype amino acid, i.e. the amino acid type at the mutation position in the x-ray crystal structure.
- `mutant_aa`: Amino acid type of the mutated protein at the mutation position.
- `StaB0_pred`: Prediction from StaB-ddG zero-shot (Deng et al., 2025), a deep learning predictor of energies based on ProteinMPNN informed only by unsupervised pretraining.
- `rde_pred`: Prediction from RDE-net (Luo et al., 2023), a deep-learning based predictor of protein binding energies.
- `wt_binder1_logprob`, `wt_binder2_logprob` and `mutant_binder_logprob`, `wt_complex_logprob`, `mutant_complex_logprob`: log probabilities predicted by `ProteinMPNN` for the wild-type (wt) and mutant amino acid sequences of the binding partners (1 and 2) separately and as a complex.
- `MPNN_embedding<d>`: Embedding features from ProteinMPNN (Dauparas et al., 2022) of the mutated position computed on the complex for embedding dimensions `d` $\in$ {`1, ..., 128`}.