Homework 3: Bayesian ML & Multiclass

Due: Tuesday, November 12, 2024 at 12 pm EST (noon)

Instructions: Your answers to the questions below, including plots and mathematical work, should be submitted as a single PDF file. It's preferred that you write your answers using software that typesets mathematics (e.g.LaTeX, LyX, or MathJax via iPython), though if you need to you may scan handwritten work.

Code Submission: Please submit your solutions in the following two files provided for the coding questions: `multiclass-skeleton-code.ipynb` and `skeleton_code.py` without changing the file names. Questions marked with [coding] requires completing the corresponding functions in the `skeleton_code.py` file. The `multiclass-skeleton-code.ipynb` file import functions written in the `skeleton_code.py`. Please do not import additional libraries in `skeleton_code.py`, otherwise the autograder may crash and no credit will be given. You should compress the files into a ZIP archive in the current directory and submit the ZIP file. We also provided two test files `test_one_vs_all_classifier.py` and `test_multiclass_svm.py` to test your code. It's meant to ensure that the output shapes are correctly specified. You can use the following commands `python -m unittest test_one_vs_all_classifier.py` and `python -m unittest test_multiclass_svm.py` to run the scripts. You don't need to submit these two test files.

Bayesian Logistic Regression with Gaussian Priors

Recall from previous homework and consider a binary classification setting with input space $\mathcal{X} = \mathbb{R}^d$, outcome space $\mathcal{Y}_\pm = \{-1, 1\}$, and a dataset $\mathcal{D} = \left((x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)})\right)$. $p(y = 1 \mid x; w) = 1/(1 + \exp(-x^T w))$.

Let's consider logistic regression in the Bayesian setting, where we introduce a prior $p(w)$ on $w \in \mathbb{R}^d$.

1. For the dataset $\mathcal{D}$, give an expression for the posterior density $p(w \mid \mathcal{D})$ in terms of the negative log-likelihood function $\mathrm{NLL}_{\mathcal{D}}(\mathrm{w})$ and the prior density $p(w)$(up to a proportionality constant is fine).

2. Show that there exist a covariance matrix Σ such that MAP (maximum a posteriori) estimate for w after observing data $\mathcal{D}$ is the same as the minimizer of the regularized logistic regression function defined in Regularized Logistic Regression paragraph above, and give its value. [Hint: Consider minimizing the negative log posterior of w. Also, remember you can drop any terms from the objective function that don't depend on w. You may freely use results of previous problems.]

3. In the Bayesian approach, the prior should reflect your beliefs about the parameters before seeing the data and, in particular, should be independent on the eventual size of your dataset. Imagine choosing a prior distribution $w \sim \mathcal{N}(0, I)$. For a dataset $\mathcal{D}$ of size n, how should you choose λ in our regularized logistic regression objective function so that the ERM is equal to the mode of the posterior distribution of w (i.e. is equal to the MAP estimator).

Coin Flipping with Partial Observability

Consider flipping a biased coin where $p(z = H \mid \theta_1) = \theta_1$. However, we cannot directly observe the result z. Instead, someone reports the result to us, which we denotey by x. Further, there is a chance that the result is reported incorrectly *if it's a head.* Specifically, we have $p(x = H \mid z = H, \theta_2) = \theta_2$ and $p(x = T \mid z = T) = 1$.

4. Show that $p(x = H \mid \theta_1, \theta_2) = \theta_1\theta_2$.
5. Given a set of reported results $\mathcal{D}_r$ of size N_r, where the number of heads is n_h and the number of tails is n_t, what is the likelihood of $\mathcal{D}_r$ as a function of θ_1 and θ_2.
6. Can we estimate θ_1 and θ_2 using MLE? Explain your judgment.
7. We additionally obtained a set of clean results $\mathcal{D}_c$ of size N_c, where x is directly observed without the reporter in the middle. Given that there are c_h heads and c_t tails, estimate θ_1 and θ_2 by MLE taking the two data sets into account. Note that the likelihood is $L(\theta_1, \theta_2) = p(\mathcal{D}_r, \mathcal{D}_c \mid \theta_1, \theta_2)$.
8. Since the clean results are expensive, we only have a small number of those and we are worried that we may overfit the data. To mitigate overfitting we can use a prior distribution on θ_1 if available. Let's imagine that an oracle gave use the prior $p(\theta_1) = \text{Beta}(h, t)$. Derive the MAP estimates for θ_1 and θ_2.

Suppose our output space and our action space are given as follows: $\mathcal{Y} = \mathcal{A} = \{1, \ldots, k\}$. Given a non-negative class-sensitive loss function $\Delta : \mathcal{Y} \times \mathcal{A} \to [0, \infty)$ and a class-sensitive feature mapping $\Psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$. Our prediction function $f : \mathcal{X} \to \mathcal{Y}$ is given by

$$f_w(x) = \arg\max_{y \in \mathcal{Y}} \langle w, \Psi(x, y) \rangle .$$

For training data $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, let $J(w)$ be the ℓ_2-regularized empirical risk function for the multiclass hinge loss. We can write this as

$$J(w) = \lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} \max_{y \in \mathcal{Y}} \left[\Delta\left(y_i, y\right) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle\right]$$

for some $\lambda > 0$.

9. Show that $J(w)$ is a convex function of w. You may use any of the rules about convex functions described in our notes on Convex Optimization, in previous assignments, or in the Boyd and Vandenberghe book, though you should cite the general facts you are using. [Hint: If $f_1, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ are convex, then their pointwise maximum $f(x) = \max\{f_1(x), \ldots, f_m(x)\}$ is also convex.]
10. Since $J(w)$ is convex, it has a subgradient at every point. Give an expression for a subgradient of $J(w)$. You may use any standard results about subgradients, including the result from an earlier homework about subgradients of the pointwise maxima of functions. (Hint: It may be helpful to refer to $\hat{y}_i = \arg\max_{y \in \mathcal{Y}} \left[\Delta\left(y_i, y\right) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle\right]$.)
11. Give an expression for the stochastic subgradient based on the point (x_i, y_i).
12. Give an expression for a minibatch subgradient, based on the points $(x_i, y_i), \ldots, (x_{i+m-1}, y_{i+m-1})$.

Hinge Loss is a Special Case of Generalized Hinge Loss

Let $\mathcal{Y} = \{-1, 1\}$. Let $\Delta(y, \hat{y}) = \mathbb{1}y \neq \hat{y}$. If $g(x)$ is the score function in our binary classification setting, then define our compatibility function as

$$\begin{aligned} h(x, 1) &= g(x)/2 \\ h(x, -1) &= -g(x)/2. \end{aligned}$$

Show that for this choice of h, the multiclass hinge loss reduces to hinge loss:

$$\ell\left(h, (x, y)\right) = \max_{y' \in \mathcal{Y}} \left[\Delta\left(y, y'\right)) + h(x, y') - h(x, y)\right] = \max\left\{0, 1 - yg(x)\right\}$$

In this problem we will work on a simple three-class classification example. The data is generated and plotted for you in the skeleton code.

One-vs-All (also known as One-vs-Rest)

First we will implement one-vs-all multiclass classification. Our approach will assume we have a binary base classifier that returns a score, and we will predict the class that has the highest score.

13. [coding] Complete the methods `fit`, `decision_function` and `predict` from `OneVsAllClassifier` in `skeleton_code.py`. Following the `OneVsAllClassifier` code is a cell that extracts the results of the fit and plots the decision region. You can have a look at it first to make sure you understand how the class will be used.

14. Include the results of the test cell in `multiclass-skeleton-code.ipynb` in your submission.

Multiclass SVM

In this question, we will implement stochastic subgradient descent for the linear multiclass SVM, as described in class and in this problem set. We will use the class-sensitive feature mapping approach with the "multivector construction", as described in the multiclass lecture.

15. [coding] Complete the function `featureMap` in `skeleton_code.py`.

16. [coding] Complete the function `sgd` in `skeleton_code.py`.

17. [coding] Complete the methods `subgradient`, `decision_function` and `predict` from the class `MulticlassSVM` in `skeleton_code.py`.

18. Following the multiclass SVM implementation, we have included another block of test code in `multiclass-skeleton-code.ipynb`. Make sure to include the results from these tests in your assignment.