

Algorithms, Regularization, and Optimization

Caleb Leedy

May 28, 2025

Overview

- There was a lot of different material in the readings this week.
- This presentation is supposed to help facilitate discussion and deepen understanding.
- It is not meant to be a complete overview.
- The course is very applied. I believe that this means that we need to implement and extend most of the results to have understanding of the topics.

Outline

1. Algorithms (from a statistics point of view)
2. Regularization
3. Optimization
4. Python on Nova

Algorithms

- Linear classifier
- k Nearest Neighbor (kNN)

Linear Classifier Algorithm

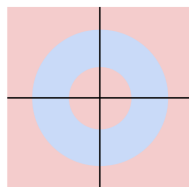
- Consider the setup of a linear classifier for a two-category problem.
- The problem is to identify the class boundaries from the sample.
- How would we describe something similar in statistics?

Class 1:

$1 \leq L2 \text{ norm} \leq 2$

Class 2:

Everything else



Linear Classifier Algorithm

- We are already familiar with a similar algorithm from statistics: logistic regression.
- For logistic regression, we assume an independent Bernoulli response model with the probability of being part of Class 1 denoted as $\pi_i(\mathbf{X})$, for covariates \mathbf{X} .
- Then the loglikelihood function is:

$$\ell(\mathbf{X}, y) = \sum_{i=1}^n \{y_i \log \pi_i(\mathbf{X}_i) + (1 - y_i) \log(1 - \pi_i(\mathbf{X}_i))\}.$$

where y_i is an indicator if i is part of Class 1 and n is the sample size.

Linear Classifier Algorithm

- This is generalized in the machine learning context to a Categorical distribution (instead of Bernoulli) to account for multiple classes.
- The functional form of $\pi_i(\mathbf{X})$ can sometimes be changed to address different forms of non-linearity.
- Sometimes the word “probabilities” is used to describe the output of logistic function. There is no reference to an underlying statistical model, so we have to be careful about what these “probabilities” mean.

k Nearest Neighbor Algorithm

- In this algorithm, you use a chosen distance function to find the closest element in the sample for prediction.
- How would we describe this problem in statistics?

k Nearest Neighbor Algorithm

- In this algorithm, you use a chosen distance function to find the closest element in the sample for prediction.
- How would we describe this problem in statistics?
- Hot-deck imputation!

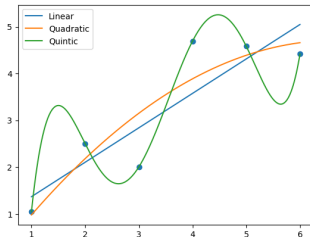
k Nearest Neighbor Algorithm

- There is a lot of literature about hot-deck imputation (See Andridge and Little (2010) for an early literature review). In particular we have,
 - Kim and Fuller (2004) for variance estimation, and
 - Yang and Kim (2018) for a theoretical framework for kNN.

Regularization

Regularization

- Consider the following setup:
 $x_i \stackrel{iid}{\sim} N(0, 1)$, $\varepsilon_i \sim N(0, 1)$ and
 $y_i = x_{1i} + \varepsilon_i$.
- If $i = \{1, 2, 3, 4, 5, 6\}$, what is the best model to use?



Regularization

- With so many other covariates, we get a really good fit for our sample, but a really bad predictor because we are using random noise to try to predict random noise.
- Regularization aims to find the signal within the noise.
- It reduces "overfitting".

Regularization

- From a statistics perspective, the L2 penalty is equivalent to ridge regression (the MAP of a Bayesian normal model for Y with a normal prior for β .)
- The L1 penalty is equivalent to the MAP of a Bayesian normal model for Y with Laplace priors on β .

Optimization

Optimization

- Optimization algorithms are procedures that minimize a (convex) loss function.

Optimization

- Gradient Descent
- Newton's Method

Gradient Descent

- The gradient descent algorithm to minimize a function $f(\mathbf{x})$ is defined to be the following:
 1. Choose an initial starting value \mathbf{x}_0 and step size s .
 2. For $t = 1, 2, 3, \dots$, update the value with:

$$\mathbf{x}_t = \mathbf{x}_{t-1} - s \nabla f(\mathbf{x})$$

3. Stop according to some stopping criteria. (Typically $|f(x_t) - f(x_{t-1})| < \varepsilon$ or $\|x_t - x_{t-1}\| < \varepsilon$.)

Justification of Gradient Descent

- Using a Taylor Expansion we can see that

$$f(\mathbf{x}^*) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{x}^* - \mathbf{x}) + \frac{\nabla^2 f(\mathbf{x})}{2} \|\mathbf{x}^* - \mathbf{x}\|^2.$$

- If we set the Hessian $H(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = s^{-1}I$ and notice that at the optimum $\nabla f(\mathbf{x}^*) = 0$, then we have

$$\begin{aligned} 0 &= \nabla f(\mathbf{x}) + s^{-1}(\mathbf{x}^* - \mathbf{x}) \\ \Rightarrow \mathbf{x}^* &= \mathbf{x} - s \nabla f(\mathbf{x}). \end{aligned}$$

Gradient Descent

- Choice of step size is important. If too small the algorithm will not reach the solution. If too big, the algorithm will not converge.
- In the last 15 years, there has been much research to determining a good step size. The algorithms Adam, AdaGrad, and RMSProp, adjust the step size for each iteration.

Newton's Method

- Without the approximation of $\nabla^2 f(x) = s^{-1}I$, we get the optimal solution to be

$$\mathbf{x}^* = \mathbf{x} - H(\mathbf{x})^{-1} \nabla f(\mathbf{x}).$$

- This is Newton's Method (or Newton-Raphson algorithm).

Comparison between Gradient Descent and Newton's Method

Gradient Descent	Newton's Method
Needs more iterations	Needs to start close enough to the solution
Only requires the gradient	Requires the gradient and Hessian

Python on Nova

Setting up the environment

- 0.(a) Start ISU VPN (if off campus).
- 0.(b) `ssh` into Nova using the command line or PuTTY if this is the first time using Nova.
 - 1. Log into Nova OnDemand.
 - 2. Start a Nova Desktop session. (Go to the Interactive Apps tab.)

Setting up the environment

3. In the Nova Desktop interactive session, open the terminal emulator via the Applications menu in the upper left corner and enter the following commands.

```
# 1. Load the Python module
module load python/3.11.9-i2aasxp

# 2. Go to project directory
cd /work/LAS/zhuz-lab/<USERNAME>/<DIRNAME>

# 3. Create virtual environment
python3 -m venv venv

# 4. Source virtual environment
source venv/bin/activate
```

Setting Up the Environment

```
# 5. Install all needed packages AND ipykernel
python -m pip install torch torchvision \
    torchaudio --index-url https://download.pytorch.org/whl/cu118
python3 -m pip install matplotlib
python3 -m pip install ipykernel

# 6. Create the kernel
python3 -m ipykernel install --user --name "<KERNELNAME>"

# 7. Deactivate virtual environment
deactivate

# 8. (Optional) Add soft link to home
cd
ln -s /work/LAS/zhuz-lab/<USERNAME>/<DIRNAME> <DIRNAME>
```

Setting up Nova

4. Go back to Nova OnDemand and start a Jupyter Lab session. (Under the Interactive Apps tab.)
5. Wait until the Connect to Jupyter button appears. Then click it.
6. When the Jupyter Lab launcher appears click the `KERNELNAME` notebook.




Updating Nova

- To add additional packages, use the following code in the Nova Desktop terminal,

```
module load python/3.11.9-i2aasxp
cd /work/LAS/zhuz-lab/<USERNAME>/<DIRNAME>
source venv/bin/activate
python3 -m pip install <PACKAGENAME>
deactivate
```

Thank You!

References I

-  Andridge, Rebecca R and Roderick JA Little (2010). “A review of hot deck imputation for survey non-response”. In: *International statistical review* 78(1), pp. 40–64.
-  Kim, Jae Kwang and Wayne Fuller (2004). “Fractional hot deck imputation”. In: *Biometrika* 91(3), pp. 559–578.
-  Yang, Shu and Jae Kwang Kim (2018). “Integration of survey data and big observational data for finite population inference using mass imputation”. In: *arXiv preprint arXiv:1807.02817*.