Yale University
Department of Statistics and Data Science

STATISTICS 365/565

Issued: 02/24/2021 **Due:** 03/05/2021

Notes: Probabilistic modeling

Notation: $[k] = \{1, 2, \ldots, k\}$. For a matrix $A \in \mathbb{R}^{m \times n}$ we will let $A_{(i,:)}$ denote the i^{th} row and $A_{(:,j)}$ denote the j^{th} column. **Both will be treated as column vectors.**

1 Problem 1: Multi-class Classification and MLE

In class we have mostly discussed binary classification. However, very often we wish to choose between many different classes, not just two. That leads to multi-class classification where we still have features $x_i \in \mathcal{X}$, but we take $y_i \in [k]$ to take on one of k different values.

a) Recall the optimal decision rule for binary classification optimizes $R(f) = \mathbb{E}[\mathbb{1}(f(x) \neq y)]$. Now compute the optimal classification rule for multi-class by optimizing the same objective $R(f)$. That is, what is $\text{argmin}_f \mathbb{E}[\mathbb{1}(f(x) \neq y)]$.

use argmax

MLE and negative log-likelihood loss In class we derived the MLE optimization problem for logistic regression. That applies to a binary classification problem. We may consider the following conditional model for multi-class classification where we take our data $Z = \{x_i, y_i\} \subset \mathbb{R}^d \times [k]$. That is $x_i \in \mathbb{R}^d$ and $y_i \in [k]$.

$$\mathbb{P}(y_i = j | x_i) = \frac{\exp(x_i^T \beta_j)}{\sum_{m=1}^k \exp(x_i^T \beta_m)}$$

Note that our parameter is now a set of vectors for each class: $\beta_1, \beta_2, \ldots, \beta_k$. As an example, we quickly see that logistic fits into this model.

$$\mathbb{P}(y_i = 0 | x_i) = \frac{\exp(x_i^T \beta_0)}{\exp(x_i^T \beta_0) + \exp(x_i^T \beta_1)}$$
$$\mathbb{P}(y_i = 1 | x_i) = \frac{\exp(x_i^T \beta_1)}{\exp(x_i^T \beta_0) + \exp(x_i^T \beta_1)}$$

By taking $\beta = \beta_1 - \beta_0$ the above will look more familiar.

A more compact way to write the above is to take $y_i \in \{0, 1\}^k$. In words we mean that y_i is a k dimensional vector that takes on only the values 0 or 1.[1] An equivalent formulation of the above conditional probability is then

$$\mathbb{P}([y_i]_{(j)} = 1, \text{ and all for all } \ell \neq j \ [y_i]_{(\ell)} = 0 | x_i) = \frac{\exp(x_i^T \beta_j)}{\sum_{m=1}^k \exp(x_i^T \beta_m)}$$

[1] In our formulation we assume that only a single entry of y_i is non-zero. However, that is not always the cases if there are overlapping categories.

The equivalence is just to get you to think about that probability taking on arbitrary values. That first part of the question is a "notational" question to yield a "simple" form of the log-likelihood as shown here: https://pytorch.org/docs/stable/generated/torch.nn.NLLLoss.html, and here https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html. I'm more concerned with you getting the form of the log-likelihood correct.

b) Take $y_i \in [k]$. Show that the following holds and then compute the log-likelihood of the β parameters

$$\mathbb{P}(y_i|x_i) = \frac{\exp(x_i^T \beta_{y_i})}{\sum_{m=1}^{k} \exp(x_i^T \beta_m)}$$

For the above, you just need to show that the conditional probability defines the same model as in the beginning of the problem statement.

c) Given we know $\mathbb{P}(y = m|x)$ in the multinomial model. Compute the optimal decision rule in this setting. Give an intuitive explanation. Note that your solution will involve an argmax. You will also be graded on the simplicity of your solution. You may ask the teaching staff if you have found an adequately simple solution.

in neural network

d) More generally, given $x_i \in \mathbb{R}^d$ and $y_i \in [k]$ we can have $0 \leq p_\ell(x_i) \leq 1$ where $\sum_{\ell=1}^{k} p_\ell(x_i) = 1$ and

$$\mathbb{P}(y_i|x_i) = \prod_{\ell=1}^{k} p_\ell(x_i)^{\mathbb{1}(y_i=\ell)}$$

l: a certain class

These probability are now arbitrary functions[2]. Verify that the negative log-likelihood loss corresponding to example i is $-\log(p_{y_i}(x_i))$. This is called the negative log-likelihood loss.

Note: Often, rather than considering the loss above, you'll consider functions $f_j(x_i)$ for $j \in [k]$ and

$$p_\ell(x_i) = \frac{\exp(f_\ell(x_i))}{\sum_{j=1}^{k} \exp(f_j(x_i))}$$

The above operation is often called the soft-max. The corresponding negative log-likelihood loss associated with example x_i and label $y_i \in [k]$ is

soft max function: the probabilities/confidence that a certain set of features belongs to a certain class.

$$-f_{y_i}(x_i) + \log\left(\sum_{j=1}^{k} \exp(f_j(x_i))\right)$$

In practice, the softmax function is used in tandem with the negative log-likelihood (NLL).

This loss is often called the cross-entropy loss. That is,

$$\ell([f_1(x), f_2(x), \ldots, f_k(x)], y) = -f_y(x) + \log\left(\sum_{j=1}^{k} \exp(f_j(x))\right)$$

functions fj(xi) are computed from the forward propagation of the network
f(xi) is a vector containing the class scores for a single example i, i.e, the output of the network.
Thus f l(xi) is an element for a certain class l in all k classes.

2 Problem 2: Generative modeling

total loss: sum over loss to correct class of all example i

In this problem we will consider the following model.

$$y = \begin{cases} 1 & \text{w.p. } \pi_1 \\ 0 & \text{w.p. } 1 - \pi_1. \end{cases}$$

[2] Almost arbitrary, they still need to be between 0 and 1 and they also need to sum to 1.

where $w.p.$ means "with probability." Let $\mathbb{R}^d \ni x = y \times w + (1-y) \times v$ where w and v are independent of y and $w \sim N(\mu_1, \Sigma_1)$ and $v \sim N(\mu_0, \Sigma_0)$. Recall that a normal random vector with mean μ and covariance Σ is denoted $N(\mu, \Sigma)$ and has pdf

$$f(x) = \frac{1}{(2\pi^d \det(\Sigma))^{1/2}} \exp\left(-\frac{(x-\mu)^T \Sigma^{-1} (x-\mu)}{2}\right)$$

a) Conditioned on $y = 1$, what is the probability density function (pdf) of x?

b) Conditioned on $y = 0$, what is the pdf of x?

c) Recall that the $\widehat{f}(x)$ that minimizes the 0/1 loss is $\widehat{f}(x) = \mathbb{1}(\eta(x) > 0.5)$ where $\eta(x) = \mathbb{P}(y = 1|x)$. Also recall that $x \in \mathbb{R}^p$.

a) Compute $\mathbb{P}(y = 1|x)$

b) Compute $\widehat{f}$. Note that $\mathbb{1}(\eta(x) > 0.5) = \mathbb{1}(\eta(x) > 1 - \eta(x))$ where recalling that $1 - \eta(x) = \mathbb{P}(y = 0|x)$ should make the problem easier. Your solution should take the form

$$\mathbb{1}\left(v^T x + x^T A x > \tau\right)$$

for some correct choices of v, τ, and A.

c) Write down the above in the special case where $p = 1$. Explain why this is called Quadratic Discriminant Analysis (QDA).

d) Compute $\widehat{f}$ for the special case where $\Sigma_1 = \Sigma_0$. Your solution should take the form

$$\mathbb{1}\left(\gamma^T x > \varepsilon\right)$$

for some correct choices of γ and ε.

e) Write down the above special case for $p = 1$. Explain why this is called Linear Discriminant Analysis (LDA).

Remark: In practice you can't use this since you do not know the values of the matrices Σ nor the values of the vectors μ. One approach is to estimate these from the data. That is for the data with label $Y = 1$, compute Σ_1 and μ_1. Similarly for $Y = 0$. Then you would plug in the appropriate values. Such methods are called *plug-in* estimators since you plug-in your estimates into the $\widehat{f}$. However, in the case of LDA this is impractical, since we just want to estimate a linear function, but go about it in a round about way.

3 Problem 3: Margin of a linear classifier

In class we have talked about the margin for a linear classifier. Recall that a linear classifier takes input $x \in \mathbb{R}^d$ and decides the label based on $w^T x$ for some weight vector w. We also discussed that

超平面（*Hyperplane*）是*n*维欧氏空间中，余维度等于*1*的线性子空间。这是平面中的直线、空间中的平面之推广

it might be good to maximize that margin among the possible separating linear decision boundaries. If we take $y_i \in \{-1, +1\}$ then the margin for example i is denoted

$$\delta_i = \frac{y_i \langle x_i, w \rangle}{\|w\|_2}$$

The separating hyperplane between the two classes of decisions is the set $H = \{v \mid \langle v, w \rangle = 0\}$. The reason this is the separating hyperplane is because any point not in this set is either classified as $+1$ or -1, while any point directly on the hyperplane has unknown classification.

Problem: Show that $|\delta_i|$ is the distance from example x_i to the hyperplane. That is prove that

$$\delta_i^2 = \min_{v | v \in H} \|v - x_i\|_2^2$$

Some steps are provided below as one approach to this problem. These steps only highlight one approach. The parts themselves do not constitute sub-problems, they are just suggested steps to consider taking.

失去一般性

Part 1. Without loss of generality you can assume that $\|w\|_2 = 1$. Otherwise you can just renormalize w. Define the vector $g = (x_i^T w)w$ and the vector $e = x_i - g$. Show that for any vector $v \in H$ the vector g is actually orthogonal to v. That is that $g^T v = 0$.

Part 2. Now show that e is actually orthogonal to w. That is that $e^T w = 0$. Use this fact to deduce that $e \in H$.

Part 3. Show that for any $v \in H$ we have

$$\begin{aligned} \|v - x_i\|_2^2 &= \|v - e - g\|_2^2 \\ &= \|v - e\|_2^2 + \|g\|_2^2 \end{aligned}$$

Part 4. Using the above you can see that the optimization can be rewritten as

$$\delta_i^2 = \min_{v | v \in H} \|v - e\|_2^2 + \|g\|_2^2$$

What is the value of δ_i^2? Verify that this result matches the statement of the problem above.