COMP 562 – Lecture 7

Gradients of Multiclass Logistic Regression log-likelihood

We will work this out in a pedestrian fashion and then obtain a compact expression:

$$\mathcal{LL}(B) = \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \left(\underbrace{\mathbf{x}_i^T \boldsymbol{\beta}_c}_{\text{involves only } \beta_c} - \underbrace{\log \left\{ \sum_{k=1}^{C} \exp\left\{ \mathbf{x}_i^T \boldsymbol{\beta}_k \right\} \right\}}_{\text{involves all columns of } B} \right)$$

$$\mathcal{LL}(B) = \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \left(\mathbf{x}_i^T \boldsymbol{\beta}_c \right) - \sum_{i=1}^{N} \log \left\{ \sum_{k=1}^{C} \exp\left\{ \mathbf{x}_i^T \boldsymbol{\beta}_k \right\} \right\}$$

Now we need to compute log-likelihood partial derivative with respect to $\beta_{j,c}$ (β associated with feature j and class c)

$$\frac{\partial}{\partial \beta_{j,c}} \mathcal{LL}(B) = \sum_{i=1}^{N} y_{i,c} x_{i,j} - \sum_{i=1}^{N} \frac{\partial}{\partial \beta_{j,c}} \log \left\{ \sum_{k=1}^{C} \exp\left\{ \mathbf{x}_i^T \boldsymbol{\beta}_k \right\} \right\}$$

On board we will work out

$$\frac{\partial}{\partial \beta_{j,c}} \log \left\{ \sum_{k=1}^{C} \exp\left\{ \mathbf{x}_i^T \boldsymbol{\beta}_k \right\} \right\} = \boxed{\frac{\exp\left\{ \mathbf{x}_i^T \boldsymbol{\beta}_c \right\}}{\sum_{k=1}^{C} \exp\left\{ \mathbf{x}_i^T \boldsymbol{\beta}_k \right\}}} x_{i,j}$$

Hence

$$\frac{\partial}{\partial \beta_{j,c}} \mathcal{LL}(B) = \sum_{i=1}^{N} y_{i,c} x_{i,j} - \sum_{i=1}^{N} \frac{\exp\{\mathbf{x}_i^T \boldsymbol{\beta}_c\}}{\sum_{k=1}^{C} \exp\{\mathbf{x}_i^T \boldsymbol{\beta}_k\}} x_{i,j}$$

For compactness, model's probability of class c for sample i will be denoted $\mu_{i,c}$

$$\mu_{i,c} = \frac{\exp\{\mathbf{x}_i^T \boldsymbol{\beta}_c\}}{\sum_{k=1}^{C} \exp\{\mathbf{x}_i^T \boldsymbol{\beta}_k\}}$$

and

$$\frac{\partial}{\partial \beta_{j,c}} \mathcal{LL}(B) = \sum_{i=1}^{N} y_{i,c} x_{i,j} - \sum_{i=1}^{N} \mu_{i,c} x_{i,j} = \sum_{i=1}^{N} x_{i,j} \underbrace{(y_{i,c} - \mu_{i,c})}_{\text{residual}}$$

$$\frac{\partial}{\partial \beta_{j,c}} \mathcal{LL}(B) = \sum_{i=1}^{N} \underbrace{x_{i,j}}_{\text{feature } j} \underbrace{(y_{i,c} - \mu_{i,c})}_{\substack{\text{residual in} \\ \text{predicting class } c}}$$

In words, partial derivative of log-likelihood with respect to j^{th} feature's weight for class c is inner product between the feature and disagreement between prediction and the true label

Gradient of log likelihood with respect to a column of B

$$\nabla_{\beta_c} \mathcal{LL}(B) = \sum_{i=1}^{N} (y_{i,c} - \mu_{i,c}) \mathbf{x}_i$$

Gradient of ridge regularized log-likelihood with respect to a column of B

$$\nabla_{\beta_c} \mathcal{PLL}(B) = \sum_{i=1}^{N} (y_{i,c} - \mu_{i,c}) \mathbf{x}_i - \lambda \begin{bmatrix} 0 \\ \mathbf{1}_p \end{bmatrix}$$

BigData and Stochastic Gradient

Once the number of samples becomes large iterating over all of them has diminishing returns

Stochastic gradient methods compute gradients using a portion of data called **mini-batches**

An extreme example of this is **online learning** -- data is streamed one sample at a time

Note: data is usually **randomaly permuted (shuffled)** before each iteration when using stochastic gradient methods

Updating parameters based on a small set of data if not guaranteed to monotonically improve log-likelihood

Hence, step-size cannot be chosen using line-search

Instead, step sizes for each iteration k are computed

$$\begin{aligned} t^{(k)} &= \left(t^{(k-1)}\right)^{1-\epsilon} \qquad && \epsilon \in [0,1] \\ t^{(k)} &= \frac{1}{\tau + k} && \tau > 0 \end{aligned}$$

and lead to diminishing step-size (learning rate)

Multivariate Gaussian Distribution -- Independent Case

Since z_1 and z_2 are independent we can write out the joint

$$\begin{aligned} p(z_1, z_2) &= p(z_1)p(z_2) \\ &= \frac{1}{\sqrt{2\pi\sigma_1^2}} \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left\{-\frac{1}{2\sigma_1^2} z_1^2\right\} \exp\left\{-\frac{1}{2\sigma_1^2} z_2^2\right\} \\ &= \frac{1}{\sqrt{(2\pi)^2 \sigma_1^2 \sigma_2^2}} \exp\left\{-\frac{1}{2\sigma_1^2} z_1^2 - \frac{1}{2\sigma_2^2} z_2^2\right\} \end{aligned}$$

In fact, for multiple independent Gaussian random variables $z_1, \ldots, z_p$ the joint is

$$\begin{aligned} p(z_1, \ldots, z_p) &= \prod_i p(z_i) \\ &= (2\pi)^{-p/2} \left(\prod_{i=1}^{p} \sigma_i^2\right)^{-1/2} \exp\left\{-\sum_{i=1}^{p} \frac{1}{2\sigma_i^2} z_i^2\right\} \end{aligned}$$

Multivariate Gaussian Distribution -- Dependent Case

Suppose we have p standard random variables (0 mean, unit variance)

$$z_i \sim \mathcal{N}(0,1), \qquad i = 1, \ldots p$$

and we are given a vector $\boldsymbol{\mu}$ of length p and a full-rank matrix A of size $p \times p$

What does distribution of $\mathbf{x} = A\mathbf{z} + \boldsymbol{\mu}$ look like?

Multivariate Gaussian Distribution -- Dependent Case

Suppose we have n standard random variables (0 mean, unit variance)

$$z_i \sim \mathcal{N}(0,1), \qquad i = 1, \dots p$$

and we are given a vector $\boldsymbol{\mu}$ of length n and a full-rank matrix A of size $p \times p$

Distribution of $\mathbf{x} = A\mathbf{z} + \mu$ is

$$p(\mathbf{x}) = (2\pi)^{-\frac{p}{2}} (\det \Sigma)^{-\frac{1}{2}} \exp\left\{ \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

where $\Sigma = AA^T$.

- μ is **mean** of the Gaussian
- Σ is **covariance** matrix

Maximum Likelihood Estimates of Mean and Covariance

Given data $\{\mathbf{x}_i \in \mathbb{R}^n | i = 1, \dots, N\}$ maximum likelihood estimates (MLE) of mean and covariance are:

$$\boldsymbol{\mu}^{\text{MLE}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

$$\Sigma^{\text{MLE}} = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\left(\mathbf{x}_i - \boldsymbol{\mu}^{\text{MLE}}\right) \left(\mathbf{x}_i - \boldsymbol{\mu}^{\text{MLE}}\right)^T}_{\text{a matrix of size } p \times p}$$

Dimensionality

- $\boldsymbol{\mu}^{\text{MLE}}$ is of same dimension as a single data point $p \times 1$.
- Σ^{MLE} is a matrix of size $p \times p$

Note that $\mathbf{x}\mathbf{x}^T$ and $\mathbf{x}^T\mathbf{x}$ are not the same, former is a matrix, latter is a scalar

Generative vs Discriminative Approaches to Classification

Thus far, we posed classification problems in terms of learning conditional probabilities of labels y given features $\mathbf{x}$

$$p(y|\mathbf{x}, \theta)$$

and we optimized **conditional** log-likelihood

$$\mathcal{LL}(\theta|\mathbf{y}, X) = \sum_i \log \underbrace{p(y_i|\mathbf{x}_i, \theta)}_{\text{conditional probability}}$$

We did not care about how features $\mathbf{x}$ were distributed

Our aim was to increase probability of labels given features

This approach to learning is called **discriminative** -- we learn to discriminate between different classes

Generative vs Discriminative Approaches to Classification

Generative models describe all of the data

$$p(y, \mathbf{x}|\theta)$$

and optimize **joint** log-likelihood

$$\mathcal{LL}(\theta|\mathbf{y}, X) = \sum_i \log \underbrace{p(y_i, \mathbf{x}_i|\theta)}_{\text{joint probability}} = \sum_i \left[\log \underbrace{p(y_i|\mathbf{x}_i, \theta)}_{\text{conditional probability}} + \log \underbrace{p(\mathbf{x}_i|\theta)}_{\text{marginal probability}} \right]$$

In this setting, the log-likelihood can be improved by:

1. Increasing conditional probability of labels given features $p(y_i|\mathbf{x}_i, \theta)$
2. Increasing probability of features $p(\mathbf{x}_i|\theta)$

However, given such a model we can describe how the data as a whole -- both features and labels -- were generated

This approach to learning is called **generative**

Generative Models for Classification

There are two ways to factorize joint probability of labels and features

$$p(y, \mathbf{x}|\theta) = p(y|\mathbf{x}, \theta)p(\mathbf{x}|\theta) = p(\mathbf{x}|y, \theta)p(y|\theta)$$

The second one given us a simple process to *GENERATE* data:

1. First select label according $p(y|\theta)$, say it was c
2. Now generate features $p(\mathbf{x}|y = c, \theta)$

Once we have such a model, to *CLASSIFY* new data, we can obtain the conditional probability $p(y|\mathbf{x})$ using Bayes rule

$$p(y = c|\mathbf{x}) = \frac{p(y = c, \mathbf{x}|\theta)}{p(\mathbf{x}|\theta)} = \frac{p(y = c|\theta)p(\mathbf{x}|y = c, \theta)}{\sum_k p(y = k|\theta)p(\mathbf{x}|y = k, \theta)}$$

and we can predict label for a new feature vector $\mathbf{x}$

Generative Models for Classification -- Prediction

If we are only interested in predicting the most likely class -- rather than computing probabilities -- we can simplify math a bit by observing

$$p(y = c|\mathbf{x}) = \frac{p(y = c|\theta)p(\mathbf{x}|y = c, \theta)}{\underbrace{\sum_k p(y = k|\theta)p(\mathbf{x}|y = k, \theta)}_{\text{does not depend on c}}}$$

Hence

$$p(y = c|\mathbf{x}) \propto p(y = c|\theta)p(\mathbf{x}|y = c, \theta)$$

and

$$\begin{aligned}\underset{c}{\operatorname{argmax}}\, p(y = c|\mathbf{x}) &= \underset{c}{\operatorname{argmax}}\, p(y = c|\theta)p(\mathbf{x}|y = c, \theta) \\ &= \underset{c}{\operatorname{argmax}}\, \log p(y = c|\theta) + \log p(\mathbf{x}|y = c, \theta)\end{aligned}$$