COMP 562 – Lecture 2

Commonly Used Discrete Distributions -- Bernoulli

Bernoulli distributed random variable X will be denoted as

$$X \sim \text{Bernoulli}(\theta)$$

State space is $\{0, 1\}$ -- think coin toss, parameter $\theta \in [0, 1]$ specifies probability of one of the outcomes

$$p(X = 1|\theta) = \theta$$
$$p(X = 0|\theta) = 1 - \theta.$$

Note that we can write this out more compactly as:

$$p(X = x|\theta) = \theta^x(1 - \theta)^{1-x}$$

Commonly Used Discrete Distributions -- Categorical

Categorical distribution is a generalization of Bernoulli to more than two outcomes -- rollin a k-sided die

$$X \sim \text{Categorical}(\theta_1, \theta_2, \ldots, \theta_{k-1})$$

State space is $\{1, 2, 3, \ldots, k\}$. Parameters $\theta_1, .. \theta_{k-1}$ specify probability of outcomes $1, \ldots k - 1$

$$p(X = 1|\theta_1, .. \theta_{k-1}) = \theta_1$$
$$p(X = 2|\theta_1, .. \theta_{k-1}) = \theta_2$$
$$\ldots$$
$$p(X = k - 1|\theta_1, .. \theta_{k-1}) = \theta_2$$
$$p(X = k|\theta_1, .. \theta_{k-1}) = 1 - \sum_{i=1}^{k-1} \theta_{k-1} = \theta_k$$

We note that θ_k is not a parameter but rather computed from parameters for convience

Commonly Used Discrete Distributions -- Binomial and Multinomial

Binomial and Multinomial distributions are generalizations of Bernoulli and Categorical distributrions

Instead of a single trial, a coin toss or die roll, we consider outcomes across multiple trials, multiple coin tosses and die rolls

$$X \sim \text{Binomial}(n, \theta)$$

$$p(X = k|n, \theta) = \binom{n}{k} \theta^k (1 - \theta)^{n-k}$$

where

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

A random variable distributed according to Multinomial distribution is a vector of counts

For example, count of 1s, 2s, 3s, 4s, 5s, 6s observed after multiple six dies rolls

$$X = (X_1, X_2, \ldots, X_k) \sim \text{Multinomial}(n, \theta)$$

$$p(X = \mathbf{x}|n, \theta) = \binom{n}{x_1 \ \ldots \ x_k} \prod_{j=1}^{k} \theta_j^{x_j}$$

where

$$\binom{n}{x_1 \ \ldots \ x_k} = \frac{n!}{x_1 ! x_2 ! \cdots x_k !}$$

Commonly Used Continuous Distributions -- Gaussian

X is distributed according to normal (or Gaussian) distribution with mean μ and variance σ^2

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

$$p(X = x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

Note that the variance is σ^2 and standard deviation is σ

Commonly Used Continuous Distributions -- Laplace

X is distributed according to Laplace distribution with location μ and scale b

$$X \sim \text{Laplace}(\mu, b)$$

$$p(X = x|\mu, b)) = \frac{1}{2b}\exp\left\{-\frac{|x-\mu|}{b}\right\}$$

Location and scale are analogous to mean and variance in normal distribution

Note:for Laplace distibution: mean = μ and variance is = $2b^2$

Parameters and Likelihood Function

Each of the coin tosses can be thought of as a realization of a random variable

Hence we can write probability of the data $\mathbf{x}$

$$p(\mathbf{x}|\theta) = \prod_{i=1}^{N} p(X = x_i|\theta) = \prod_{i=1}^{N} \theta^{x_i}(1-\theta)^{1-x_i}$$

Pluging in different θs give different probability of the data

Q: Could this help us figure out what the next toss could be?

Parameters and Likelihood Function

Hence, we can try to find parameter θ for which $p(\mathbf{x}|\theta)$ is the largest

$p(\mathbf{x}|\theta)$ can be seen as a function of **parameter** θ

This function is called *likelihood*

$$\mathcal{L}(\theta|\mathbf{x}) = p(\mathbf{x}|\theta)$$

θ which results in the largest likelihood is called **maximum-likelihood estimate**

In many cases, learning is nothing more than maximizing likelihood

Log-Likelihood

Maximization of likelihood can be tricky when datasets are large, computing product of probabilities, by definition smaller than 1, can easily underflow

Typically, we maximize likelihood by finding maxima of log-likelihood, the location of the maximum's of these two functions coincide, and the only difference is that we avoid numerical problems

$$\log \mathcal{L}(\theta|\mathbf{x}) = \log p(\mathbf{x}|\theta) = \log \prod_i p(x_i|\theta) = \sum_i \log p(x_i|\theta)$$

In general, we will compute log probabilities and only convert them to probabilities when we need to perform marginalization

Maximizing Likelihood

Log-Likelihood in detail:

$$\log \mathcal{L}(\theta|\mathbf{x}) = \sum_i \log p(x_i|\theta)$$

We plug in our Bernoulli distribution

$$p(x_i|\theta) = \theta^{x_i}(1-\theta)^{1-x_i}$$

and it's log is

$$\log p(x_i|\theta) = x_i \log \theta + (1-x_i)\log(1-\theta)$$

Putting it all together

$$\log \mathcal{L}(\theta|\mathbf{x}) = \sum_i \left[x_i \log \theta + (1-x_i)\log(1-\theta)\right]$$

Maximizing Likelihood

Our coin toss example:

Data:

$$\mathbf{x} = \{0, 1, 0, 0, 1, 0, 1, 0, 1, \dots\}$$

Log-Likelihood:

$$\log \mathcal{L}(\theta|\mathbf{x}) = \log p(\mathbf{x}|\theta) = \log \prod_i p(x_i|\theta) = \sum_i \log p(x_i|\theta)$$

Maximum likelihood estimate:

$$\theta^{\mathrm{ML}} = \underset{\theta}{\operatorname{argmax}} \log \mathcal{L}(\theta|\mathbf{x})$$

Q: How do we find maxima/minima of functions?

Finding Maximum/Minimum of log-Likelihood

Coin toss example again:

$$\log \mathcal{L}(\theta|\mathbf{x}) = \sum_i \left[x_i \log \theta + (1 - x_i) \log(1 - \theta)\right]$$

Let's compute first derivative

$$\frac{\partial}{\partial\theta}\log \mathcal{L}(\theta|\mathbf{x}) = \sum_i \left[x_i \frac{1}{\theta} + (1 - x_i)(-\frac{1}{1 - \theta})\right]$$

To find best θ we equate the derivative $\frac{\partial}{\partial\theta}\log \mathcal{L}(\theta|\mathbf{x})$ to zero and solve

$$\frac{\partial}{\partial\theta}\log \mathcal{L}(\theta|\mathbf{x}) = 0$$

Finding Maximum/Minimum of log-Likelihood

For our coin toss example this amounts to:

$$\sum_i \left[x_i \frac{1}{\theta} + (1 - x_i)\frac{-1}{1 - \theta}\right] = 0$$

$$\sum_i x_i \frac{1}{\theta} - \sum_i (1 - x_i)\frac{1}{1 - \theta} = 0$$

$$\sum_i x_i \frac{1}{\theta} = \sum_i (1 - x_i)\frac{1}{1 - \theta}$$

$$\frac{1}{\theta}\underbrace{\sum_i x_i}_{n_h} = \frac{1}{1 - \theta}\underbrace{\sum_i (1 - x_i)}_{n_t}$$

$$(1 - \theta)n_h = \theta n_t$$

$$n_h = \theta(n_t + n_h)$$

$$\theta = \frac{n_h}{n_t + n_h}$$

Anti-climactic? Reassuring? From the first principles?

COMP 562 – Lecture 3

Finding μ^{MLE} of Gaussian Distribution

We left as an exercise a problem to come up with a maximum likelihood estimate for parameter μ of a Gaussian distribution

$$p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

So we will do that now

Likelihood function is

$$\mathcal{L}(\mu,\sigma^2|\mathbf{x}) = \prod_{i=1}^{N} p(x_i|\mu,\sigma^2) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x_i-\mu)^2}$$

Log-likelihood function is

$$\log \mathcal{L}(\mu,\sigma^2|\mathbf{x}) = \log \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x_i-\mu)^2} = \sum_{i=1}^{N} \left[-\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}(x_i-\mu)^2 \right]$$

Finding μ^{MLE} of Gaussian Distribution

Our recipe is:

1. Take the function you want to maximize:

$$f(\mu) = \sum_{i=1}^{N} \left[-\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}(x_i-\mu)^2 \right]$$

1. Compute its first derivative: $\frac{\partial}{\partial\mu} f(\mu)$
2. Equate that derivative to zero and solve: $\frac{\partial}{\partial\mu} f(\mu) = 0$

The first derivative is

$$\frac{\partial}{\partial \mu} f(\mu) = \sum_{i=1}^{N} \left[\frac{1}{\sigma^2} (x_i - \mu) \right]$$

We equate it to zero and solve

$$\sum_{i=1}^{N} \left[\frac{1}{\sigma^2} (x_i - \mu) \right] = 0$$

$$\frac{\sum_{i=1}^{N} x_i}{N} = \mu$$

Finding ${\sigma^2}^{MLE}$ of Gaussian Distribution

Our recipe is:

1. Take the function you want to maximize:

$$f(\sigma^2) = \sum_{i=1}^{N} \left[-\frac{1}{2} \log 2\pi\sigma^2 - \frac{1}{2\sigma^2} (x_i - \mu)^2 \right]$$

1. Compute its first derivative: $\frac{\partial}{\partial \sigma^2} f(\sigma^2)$
2. Equate that derivative to zero and solve: $\frac{\partial}{\partial \sigma^2} f(\sigma^2) = 0$

$$f(\sigma^2) = \sum_{i=1}^{N} \left[-\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}(x_i - \mu)^2 \right] = -\frac{N}{2}\log 2\pi - \frac{N}{2}\log \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{N} \left[(x_i - \mu)^2 \right]$$

The first derivative is

$$\frac{\partial}{\partial \sigma^2} f(\sigma^2) = -\frac{N}{2\sigma^2} - \left(\frac{1}{2} \sum_{i=1}^{N} \left[(x_i - \mu)^2 \right] \right) \frac{\partial}{\partial \sigma^2} \left(\frac{1}{\sigma^2} \right)$$

$$= -\frac{N}{2\sigma^2} - \left(\frac{1}{2} \sum_{i=1}^{N} \left[(x_i - \mu)^2 \right] \right) \left(-\frac{1}{(\sigma^2)^2} \right) = \frac{1}{2\sigma^2} \left(\frac{1}{\sigma^2} \sum_{i=1}^{N} \left[(x_i - \mu)^2 \right] - N \right)$$

Which, if we rule out $\sigma^2 = 0$, is equal to zero only if

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} \left[(x_i - \mu)^2 \right]$$

Please Verify both μ^{MLE} and ${\sigma^2}^{MLE}$ using seconed derivative test

Linear Regression

Formally, we would write

$$y = \beta_0 + \sum_j x_j \beta_j + \epsilon$$
$$\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$$

or more compactly

$$y|\mathbf{x} \sim \mathcal{N}\left(\beta_0 + \sum_j x_j \beta_j, \sigma^2 \right)$$

Notice that the function is linear in the parameters $\beta = (\beta_0, \beta_1, \ldots, \beta_p)$, not necessarily in terms of the covariates

Linear Regression

Probability of target variable y

$$p(y|\mathbf{x}, \beta_0, \beta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2} \left(y_i - \underbrace{(\beta_0 + \sum_j x_j \beta_j)}_{\text{mean of the Gaussian}} \right)^2 \right\}$$

In the case of the 6th grader's height, we made **the same** prediction for any other 6th grader (58.5 inches)

In our COMP 562 grade example, we compute a potentially different mean for every student

$$\beta_0 + \beta_{\text{COMP410}} * \text{COMP410} + \beta_{\text{MATH233}} * \text{MATH233} + \beta_{\text{STOR435}} * \text{STOR435} + \beta_{\text{beers}} * \text{beers}$$

Linear Regression -- Likelihood

We start by writing out a likelihood for linear regression is

$$\mathcal{L}(\beta_0, \beta, \sigma^2|\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{N} p(y|\mathbf{x}, \beta_0, \beta, \sigma^2) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2} \left(y_i - (\beta_0 + \sum_j x_j \beta_j) \right)^2 \right\}$$

Log-likelihood for linear regression is

$$\log \mathcal{L}(\beta_0, \beta, \sigma^2|\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \left[-\frac{1}{2} \log 2\pi\sigma^2 - \frac{1}{2\sigma^2} \left(y_i - (\beta_0 + \sum_j x_j \beta_j) \right)^2 \right]$$

$$= -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{N} \left(y_i - (\beta_0 + \sum_j x_{i,j} \beta_j) \right)^2 = -\frac{N}{2} \log(2\pi\sigma^2) - \frac{RSS}{2\sigma^2}$$

We will refer to expression $y_i - (\beta_0 + \sum_j x_j\beta_j)$ as **residual**, and hence **RSS** stands for **residual sum of squares** or **sum of squared errors** and is defined by

$$RSS = \sum_{i=1}^{N} \left(y_i - (\beta_0 + \sum_j x_{i,j}\beta_j) \right)^2$$

And RSS/N is called the **mean squared error** or **MSE**

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - (\beta_0 + \sum_j x_{i,j}\beta_j) \right)^2$$

Hence, maximizing log-likelihood is equivalent to minimizing RSS or MSE

Another way to see this is to consider a very simplified version of Taylor's theorem

Theorem. Given a function $f(\cdot)$ which is smooth at x

$$f(x + d) = f(x) + f'(x)d + O(d^2)$$

In words, close to x function $f(\cdot)$ is very close to being a linear function of d

$$f(x + d) = f(x) + f'(x)d$$

Slope of the best linear approximation is $f'(x)$, i.e., $f'(x)$ tells us in which direction function grows

- Gradient Ascent\Descent: Choose initial $\theta^{(0)} \in \mathbb{R}^n$, repeat:
  $$\theta^{(k)} = \theta^{(k-1)} \pm t_k . \nabla f(\theta^{(k-1)}), k = 1, 2, 3, \ldots$$
  Where t_k is the step size (learning rate) at step k
- Stop at some point using a stopping criteria (depend on the problem we are solving), for example:
  - Maximum number of iterations reached
  - $|f(\theta^{(k)}) - f(\theta^{(k-1)})| < \epsilon$

1. Use Line search Strategy

   - At each iteration, do the best you can along the direction of the gradient,
     $$t = \underset{s \geq 0}{\operatorname{argmax}} f(\theta + s . \nabla f(\theta))$$
   - Usually, it is not possible to do this minimization exactly, and approximation methods are used
   - Backtracking Line Search:
     - Choose an initial learning rate ($t_k = t_{init}$), and update your parameters
       $\theta^{(k)} = \theta^{(k-1)} \pm t_k . \nabla f(\theta^{(k-1)})$
     - Reduce learning rate $t_k = \alpha . t_{init}$, where $0 < \alpha < 1$
     - Repeat by reducing α till you see an improvmnet in $f(\theta^{(k)})$

Linear Regression -- Likelihood

We start by writing out a likelihood for linear regression is

$$\mathcal{L}(\beta_0, \beta, \sigma^2 | \mathbf{x}, \mathbf{y}) = \prod_{i=1}^{N} p(y | \mathbf{x}, \beta_0, \beta, \sigma^2) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2} \left(y_i - (\beta_0 + \sum_j x_j \beta_j) \right)^2 \right\}$$

Log-likelihood for linear regression is

$$\log \mathcal{L}(\beta_0, \beta, \sigma^2 | \mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \left[-\frac{1}{2} \log 2\pi\sigma^2 - \frac{1}{2\sigma^2} \left(y_i - (\beta_0 + \sum_j x_j \beta_j) \right)^2 \right].$$

Linear Regression -- Gradient of Log-Likelihood

Partial derivatives

$$\frac{\partial}{\partial \beta_0} \log \mathcal{L}(\beta_0, \beta, \sigma^2 | \mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} -\frac{1}{\sigma^2} \left(y_i - (\beta_0 + \sum_j x_j \beta_j) \right) (-1)$$

$$\frac{\partial}{\partial \beta_k} \log \mathcal{L}(\beta_0, \beta, \sigma^2 | \mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} -\frac{1}{\sigma^2} \left(y_i - (\beta_0 + \sum_j x_j \beta_j) \right) (-x_k) \qquad , k \in \{1, \dots, p\}$$

Hence gradient (with respect to βs)

$$\nabla \log \mathcal{L}(\beta_0, \beta, \sigma^2 | \mathbf{x}, \mathbf{y}) = \begin{bmatrix} \sum_{i=1}^{N} -\frac{1}{\sigma^2} \left(y_i - (\beta_0 + \sum_j x_j \beta_j) \right) (-1) \\ \sum_{i=1}^{N} -\frac{1}{\sigma^2} \left(y_i - (\beta_1 + \sum_j x_j \beta_j) \right) (-x_1) \\ \vdots \\ \sum_{i=1}^{N} -\frac{1}{\sigma^2} \left(y_i - (\beta_0 + \sum_j x_j \beta_j) \right) (-x_p) \end{bmatrix}$$

COMP 562 – Lecture 4

Linear Regression -- Matrix Form

A general multiple-regression model can be written as

$$y|\mathbf{x} = \beta_0 + \sum_j x_j\beta_j + \epsilon, \quad \epsilon \sim \mathcal{N}\left(, 0, \sigma^2\right)$$

Which is equivalent to

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon_i \quad for\, i = 1, \ldots, N$$

In matrix form, we can rewrite this model as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1p} \\ 1 & x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_{N1} & x_{N2} & \ldots & x_{Np} \end{bmatrix}_{N\times p+1} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_P \end{bmatrix}_{p+1\times 1} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix}_{N\times 1}$$

This can be rewritten more simply as:

$$\mathbf{y} = \mathbf{X}\beta + \epsilon$$

Linear Regression -- Closed Form Solution for β

Remember that maximizing log-likelihood is equivalent to minimizing **RSS** or **MSE**

$$RSS = \sum_{i=1}^{N}\left(y_i - (\beta_0 + \sum_j x_{i,j}\beta_j)\right)^2 = \sum_{i=1}^{N}(e_i)^2 = \mathbf{e_i}^T\mathbf{e_i} = \begin{bmatrix} e_1 & e_2 & \ldots & e_N \end{bmatrix}_{1\times N}\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix}_{N\times 1}$$

$$\begin{aligned} RSS &= (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \\ &= (\mathbf{y}^T - \beta^T\mathbf{X}^T)(\mathbf{y} - \mathbf{X}\beta) \\ &= \mathbf{y}^T\mathbf{y} - \beta^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\beta + \beta^T\mathbf{X}^T\mathbf{X}\beta \\ &= \mathbf{y}\mathbf{y}^T - 2\beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta \end{aligned}$$

Where this development uses the fact that the transpose of a scalar is the scalar i.e. $\beta^T\mathbf{X}^T\mathbf{y} = \mathbf{y}^T\mathbf{X}\beta$

To find the β that minimizes RSS, we solve the following equation:

$$\nabla_\beta \, RSS = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\beta = 0$$

The corresponding solution to this linear system of equations is called the **ordinary least squares** or **OLS** solution

$$\hat{\beta} = \beta^{\text{OLS}} = \beta^{\text{MLE}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Linear Regression -- Closed Form Solution for σ^2

Recall log-likelihoood function

$$\log \mathcal{L}(\beta_0, \beta|\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{N}\left[-\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\left(y_i - (\beta_0 + \sum_j x_{i,j}\beta_j)\right)^2\right]$$

Which can be written in matrix form

$$\log \mathcal{L}(\beta|\mathbf{y}, \mathbf{x}) = -\frac{N}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$$

Taking derivative and equating it to zero yields

$$(\sigma^2)^{\text{MLE}} = \frac{1}{N}(\mathbf{y} - \mathbf{X}\beta^{\text{MLE}})^T(\mathbf{y} - \mathbf{X}\beta^{\text{MLE}}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - (\beta_0^{\text{MLE}} + \sum_j x_{i,j}\beta_j^{\text{MLE}})\right)^2$$

Please verify $(\sigma^2)^{\text{MLE}}$ at home

- **Overfitting**: Model every minor variation in the input using highly flexible complex models
  - High variance and low bias
- **Underfitting**: Simple model that is unable to capture the true relationships in given data
  - Low variance and high bias
- **Model Selection**: Picking the right model from a variety of models of different complexity

Q: Which model overfits/underfits the data?

Bias-Variance Tradeoff

The **mean quared errors** or **MSE** may be decomposed into **bias** and **variance** components:

$$\underbrace{\mathbb{E}(y-\hat{y})^2}_{\textbf{MSE}} = \underbrace{(\mathbb{E}(\hat{y})-y)^2}_{\textbf{Bias}^2} + \underbrace{\mathbb{E}\left[(\hat{y}-\mathbb{E}(\hat{y}))^2\right]}_{\textbf{Variance}} + \underbrace{\sigma_e^2}_{\textbf{Irreducible Error}}$$

Ill-Posed Problems

Q: What happens if you are solving a linear system $Ax = y$ and there are more unknowns than equations?

In our setting -- N samples, P features -- linear regresion is ill-posed if $P > N$

Another example of ill-posed linear regression problem arises when we have two copies of the same predictors

This is a problem even if $P < N$

Ridge Regression

Adding that penalty to linear regression log-likelihood yields **ridge regresion**

$$\log \mathcal{L}(\beta_0, \beta | \mathbf{y}, \mathbf{x}) = \sum_{i=1}^{N} \left[-\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\left(y_i - (\beta_0 + \sum_j x_{i,j}\beta_j) \right)^2 \right] - \underbrace{\frac{\lambda}{2}\sum_j \beta_j^2}_{\text{ridge penalty}}$$

All those sums can get cumbersome, so we will use norms

1. ℓ_2 norm $\|\mathbf{x}\| = \sqrt{\sum_i x_i^2}$
2. ℓ_1 norm $\|\mathbf{x}\|_1 = \sum_i |x_i|$

$$\log \mathcal{L}(\beta | \mathbf{y}, \mathbf{x}) = -\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{X}\beta\|^2 \underbrace{-\frac{\lambda}{2}\|\beta\|^2}_{\text{ridge penalty}} + \text{const.}$$

Ridge Regression -- Computing Gradients

$$\log \mathcal{L}(\beta|\mathbf{y}, \mathbf{x}) = -\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{X}\beta\|^2 \underbrace{-\frac{\lambda}{2}\|\beta\|^2}_{\text{ridge penalty}} + \text{const.}$$

Computing the gradient and setting it to zero

$$\nabla_\beta \log \mathcal{L}(\beta|\mathbf{y}, \mathbf{x}) = \frac{1}{\sigma^2} X^T(\mathbf{y} - \mathbf{X}\beta) - \lambda\beta = 0$$

yields

$$\beta^{\text{MLE}} = (\mathbf{X}^T\mathbf{X} + \lambda\sigma^2\mathbf{I}_N)^{-1}(\mathbf{X}^T\mathbf{y})$$

Where $\mathbf{I}_N$ is the identity matrix of size N

Contrast this to closed form solution of linear regression

$$\beta^{\text{MLE}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Ridge Regression -- Computing Gradients

The bias/intercept coefficient β_0 is typically not regularized in a linear regression

A regularized β_0 (shrinked) may us from prevent finding the correct relationship

$$\nabla \log \mathcal{L}(\beta_0, \beta, \sigma^2|\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \sum_{i=1}^N -\frac{1}{\sigma^2}\left(y_i - (\beta_0 + \sum_j x_{i,j}\beta_j)\right)(-1) \\ \sum_{i=1}^N -\frac{1}{\sigma^2}\left(y_i - (\beta_1 + \sum_j x_{i,j}\beta_j)\right)(-x_{i,1}) - \lambda\beta_1 \\ \vdots \\ \sum_{i=1}^N -\frac{1}{\sigma^2}\left(y_i - (\beta_0 + \sum_j x_{i,j}\beta_j)\right)(-x_{i,p}) - \lambda\beta_p \end{bmatrix}$$

Note that β_0 is **not** regularized

Remember our closed form solution for ridge regression

$$\beta^{\text{MLE}} = (\mathbf{X}^T\mathbf{X} + \lambda\sigma^2\mathbf{I}_N)^{-1}(\mathbf{X}^T\mathbf{y})$$

Updating our closed form solution without regularizing β_0 will yeild

$$\beta^{\text{MLE}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\sigma^2 \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}_{N\times N}\right)^{-1}(\mathbf{X}^T\mathbf{y})$$

COMP 562 – Lecture 5

Feature Scaling -- Feature Scaling

- Idea: gradient ascent/descentalgorithm tends to work better if the features are on the **same scale**

 When contours are skewed then learning steps would take longer to converge due to oscillatory behaviour

Feature Scaling -- Centering

Center features by removing the mean

$$\mu_i = \frac{1}{N}\sum_{k=1}^{N} x_{i,k}$$

$$x_{i,j} = x_{i,j} - \mu_i$$

This makes each feature's mean equal to 0. Compute the mean first, then subtract it!

Feature Scaling -- Standardizing

Standardize centered features by dividing by the standard deviation

$$\sigma_i = \sqrt{\frac{1}{N-1}\sum_{j} x_{i,j}^2}$$

$$x_{i,j} = \frac{x_{i,j}}{\sigma_i}$$

Note that standardized features are first centered and then divided by their standard deviation

Transform your data to a distribution that has a mean of 0 and a standard deviation of 1 (z-score)

Feature Scaling -- Normalizing

Alternatively, **normalize** centered features by dividing by their norm

$$r_i = \sqrt{\sum_j x_{i,j}^2}$$

$$x_{i,j} = \frac{x_{i,j}}{r_i}$$

Note that normalized features are first centered and then divided by their norm

Normalization transforms your data into a range between 0 and 1 regardless of the data set size

Feature Scaling Benefits

1. Centering
   A. β_0 is equal to the mean of the target variable
   B. Feature weights β now tell us how much does feature's departure from mean affect the target variable
2. Standardization
   A. All the features are on the same scale and their effects comparable
   B. Interpretation is easier: βs tell us how much departure by single standard deviation affects the target variable
3. Normalization
   A. Scale of features is the same, regardles of the size of the dataset
   B. Hence weights learend on different sized datasets can be compared
   C. However, their combination might be problematic -- certainly we don't trust weights learned on few samples

Classification -- Bernoulli View

We can model a target variable $y \in \{0, 1\}$ using Bernouli distribution

$$p(y = 1|\theta) = \theta$$

We note that θ has to be in range $[0, 1]$

We cannot directly take weighted combination of features to obtain θ

We need a way to map $\mathbf{x}^T\beta \in \mathbb{R}$ to range $[0, 1]$

Some Useful Equalities Involving Sigmoid

Definition:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Recognize the alternative way to write it:

$$\sigma(z) = \frac{\exp z}{1 + \exp z}$$

Complement is just flip of the sign in the argument

$$\sigma(-z) = 1 - \sigma(z)$$

Log ratio of probability (log odds)

$$\log \frac{\sigma(z)}{\sigma(-z)} = z$$

Using Sigmoid to Parameterize Bernoulli

$$p(y = 1|\theta) = \theta$$

Sigmoid "squashes" the whole real line into range $[0, 1]$

Hence we can map weighted features into a parameter θ

$$\theta = \sigma(\beta_0 + \mathbf{x}^T\beta)$$

and use that θ in our Bernoulli

$$p(y = 1|\theta = \sigma(\beta_0 + \mathbf{x}^T\beta)) = \sigma(\beta_0 + \mathbf{x}^T\beta)$$

Logistic Regression -- Binary Classification

In logistic regression we model a binary variable $y \in \{-1, +1\}$

$$p(y = +1|\mathbf{x}, \beta_0, \beta) = \sigma\left(+(\beta_0 + \mathbf{x}^T\beta)\right)$$
$$p(y = -1|\mathbf{x}, \beta_0, \beta) = 1 - \sigma\left(-(\beta_0 + \mathbf{x}^T\beta)\right) = \sigma\left(-(\beta_0 + \mathbf{x}^T\beta)\right)$$

This is equivalent to

$$p(y|\mathbf{x}, \beta_0, \beta) = \sigma\left(y(\beta_0 + \mathbf{x}^T\beta)\right) = \frac{1}{1 + \exp\left\{-y(\beta_0 + \mathbf{x}^T\beta)\right\}}$$

Q: Does above formula work for $y \in \{0, 1\}$?

Logistic Regression -- Decision Boundary

$$p(y = 1|\mathbf{x}, \beta_0, \beta) = \sigma\left((\beta_0 + \mathbf{x}^T\beta)\right) = \frac{1}{1 + \exp\left\{-(\beta_0 + \mathbf{x}^T\beta)\right\}}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

- Suppose predict "$y = 1$" if

$$p(y = 1|\mathbf{x}, \beta_0, \beta) \geq 0.5 \rightarrow \beta_0 + \mathbf{x}^T\beta \geq 0$$

- Then predict "$y = -1$" if

$$p(y = 1|\mathbf{x}, \beta_0, \beta) < 0.5 \rightarrow \beta_0 + \mathbf{x}^T\beta < 0$$

- Hence, the decision boundary is given by $\beta_0 + \mathbf{x}^T\beta = 0$

Q: What does this decision boundary equation describe?

Logistic Regression -- Log-Likelihood

Probability of a single sample is:

$$p(y|\mathbf{x}, \beta_0, \beta) = \frac{1}{1 + \exp\{-y(\beta_0 + \mathbf{x}^T\beta)\}}$$

Likelihood function is:

$$\mathcal{L}(\beta_0, \beta|\mathbf{y}, \mathbf{x}) = \prod_i \frac{1}{1 + \exp\{-y_i(\beta_0 + \mathbf{x}_i^T\beta)\}}$$

Log-likelihood function is:

$$\log \mathcal{L}(\beta_0, \beta|\mathbf{y}, \mathbf{x}) = -\sum_i \log\{1 + \exp\{-y_i(\beta_0 + \mathbf{x}_i^T\beta)\}\}$$

Follow the same recipe as before to find βs that maximize the Log-likelihood function

COMP 562 – Lecture 6

Logistic Regression -- Log-Likelihood for $\pm$ 1 Labels

Probability of a single sample is when $y \in \{-1, +1\}$:

$$p(y|\mathbf{x}, \beta_0, \beta) = \frac{1}{1 + \exp\{-y(\beta_0 + \mathbf{x}^T\beta)\}}$$

Likelihood function is:

$$\mathcal{L}(\beta_0, \beta|\mathbf{y}, \mathbf{x}) = \prod_i \frac{1}{1 + \exp\{-y_i(\beta_0 + \mathbf{x}_i^T\beta)\}}$$

Log-likelihood function is:

$$\mathcal{LL}(\beta_0, \beta|\mathbf{y}, \mathbf{x}) = -\sum_i \log\{1 + \exp\{-y_i(\beta_0 + \mathbf{x}_i^T\beta)\}\}$$

Logistic Regression -- Log-Likelihood for $0, 1$ Labels

Probability of a single sample is when $y \in \{0, 1\}$:

$$p(y|\mathbf{x}, \beta_0, \beta) = \frac{\exp\{y(\beta_0 + \mathbf{x}^T\beta)\}}{1 + \exp\{(\beta_0 + \mathbf{x}^T\beta)\}}$$

Likelihood function is:

$$\mathcal{L}(\beta_0, \beta|\mathbf{y}, \mathbf{x}) = \prod_i \frac{\exp\{y_i(\beta_0 + \mathbf{x}_i^T\beta)\}}{1 + \exp\{(\beta_0 + \mathbf{x}^T\beta)\}}$$

Log-likelihood function is:

$$\mathcal{LL}(\beta_0, \beta|\mathbf{y}, \mathbf{x}) = \sum_i y_i(\beta_0 + \mathbf{x}_i^T\beta) - \log\{1 + \exp\{(\beta_0 + \mathbf{x}_i^T\beta)\}\}$$

Ridge Penalty and Logistic Regression

Adding ridge penalty to the logistic regression achieves

1. Shrinkage of weights -- weights no longer explode in separable case
2. Even splitting between correlated weights

Ridge regularized log-likelihood for $\pm$ 1 labels:

$$\mathcal{PLL}(\beta_0, \beta|\mathbf{y}, \mathbf{x}) = -\sum_i \log\left\{1 + \exp\left\{-y_i(\beta_0 + \mathbf{x}_i^T\beta)\right\}\right\} - \frac{\lambda}{2}\|\beta\|^2$$

Ridge regularized log-likelihood for $0, 1$ labels:

$$\mathcal{PLL}(\beta_0, \beta|\mathbf{y}, \mathbf{x}) = \sum_i y_i(\beta_0 + \mathbf{x}_i^T\beta) - \log\left\{1 + \exp\left\{(\beta_0 + \mathbf{x}_i^T\beta)\right\}\right\} - \frac{\lambda}{2}\|\beta\|^2$$

Bayesian View of Penalties

We have seen two examples of supervised models

1. Linear regression, $p(y|\mathbf{x}, \beta)$ where $y \in \mathbb{R}$
2. Logistic regression, $p(y|\mathbf{x}, \beta)$ where $y \in \{-1, +1\}$

We then utilized log-likelihoods

$$\mathcal{LL}(\beta|\mathbf{y}, X) = \sum_i \log p(y_i|\mathbf{x}_i, \beta)$$

and observed that we can add penalties to log-likelihoods

$$\mathcal{LL}(\beta|\mathbf{y}, X) + \lambda f(\beta)$$

in order to deal with ill-posedness of the problems

Bayesian View of Penalties

Given a likelihood

$$p(\text{Data}|\theta)$$

Bayesian view of models treats each parameter θ as just another random variable

This random variable has a distribution called **prior** distribution

$$p(\theta)$$

Using Bayes rule we can also compute

$$\overbrace{p(\theta|\text{Data})}^{\text{posterior}} = \frac{\overbrace{p(\text{Data}|\theta)}^{\text{likelihood}}\overbrace{p(\theta)}^{\text{prior}}}{\underbrace{p(\text{Data})}_{\text{evidence}}}$$

called **posterior** distribution

Prior encodes our beliefs **before** seing the data

Posterior reflects our updated beliefs **after** seeing the data

Bayesian View of Penalties

For example we can assume a Gaussian **prior** on β_i to our linear regression model

$$\beta_i \sim \mathcal{N}\left(0, \frac{1}{\lambda}\right), \qquad i > 0$$
$$y \sim \mathcal{N}\left(\beta_0 + \mathbf{x}^T\beta, \sigma^2\right)$$

Then posterior probability of the parameter β_i:

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

Bayesian View of Penalties

We can now try to find **Maximum-A-Posteriori (MAP)** estimate of θ

$$\arg\max_{\beta} p(\beta|\mathbf{y},\mathbf{x}) = \arg\max_{\beta} \log p(\mathbf{y}|\mathbf{x},\beta) + \log p(\beta)$$

and this is equivalent to

$$\arg\max_{\beta} p(\beta|\mathbf{y},\mathbf{x}) = \arg\max_{\beta} -\sum_{i=1}^{N} \frac{1}{2\sigma^2}(y_i - \beta_0 - \mathbf{x}_i^T\beta) - \sum_{j=1}^{p} \frac{\lambda}{2}\beta_j^2 + \text{const}$$

Solving ridge regression is equivalent to finding Maximum-A-Posteriori estimate in Bayesian linear regression with Gaussian prior on weights

Softmax

Sigmoid:

$$\sigma(z) = \frac{1}{1+\exp\{-z\}} = \frac{\exp\{z\}}{1+\exp\{z\}}$$

Softmax is a generalization of sigmoid:

$$\sigma(\mathbf{z})_j = \frac{\exp\{z_j\}}{\sum_{c=1}^{C}\exp\{z_j\}}$$

For example:

$$\sigma(\mathbf{z})_1 = \frac{\exp\{z_1\}}{\exp\{z_1\}+\exp\{z_2\}+\exp\{z_3\}}$$
$$\sigma(\mathbf{z})_2 = \frac{\exp\{z_2\}}{\exp\{z_1\}+\exp\{z_2\}+\exp\{z_3\}}$$
$$\sigma(\mathbf{z})_3 = \frac{\exp\{z_3\}}{\exp\{z_1\}+\exp\{z_2\}+\exp\{z_3\}}$$

Multiclass Logistic Regression and Softmax

We can write out probability of partcular class using softmax

$$p(y = c|\mathbf{x}, \beta_0, B) = \boxed{\frac{\exp\{\beta_{0,c} + \mathbf{x}^T\boldsymbol{\beta}_c\}}{\sum_{k=1}^{C}\exp\{\beta_{0,k} + \mathbf{x}^T\boldsymbol{\beta}_k\}}}$$

where

$$B = \left[\boldsymbol{\beta}_1\boldsymbol{\beta}_2\ldots\boldsymbol{\beta}_C\right]$$

and each $\boldsymbol{\beta}_c$ is a vector of class specific feature weights

Note that the $p(y = c|\cdots)$ is a categorical distribution over C possible states, where probabilities of each state are given by softmax

Multiclass Logistic Regression -- Log-Likelihood

1. There are N samples, each in one of C classes, and p features
2. Labels are represented using one-hot vectors y_i
3. Feature matrix X contains a column of 1s -- corresponding to the bias term
4. First row of weight matrix B are bias terms
5. $\boldsymbol{\beta}_k$ is k^{th} column of matrix B

Dimensions:

- Feature matrix : $X \to N \times (p + 1)$
- Label matrix : $Y \to N \times C$
- Weight matrix : $B \to (p + 1) \times C$

Likelihood is

$$\mathcal{L}(B|Y, X) = \underbrace{\prod_{i=1}^{N}}_{\text{samples}}\underbrace{\prod_{c=1}^{C}}_{\text{classes}}\left[\frac{\exp\{\mathbf{x}_i^T\boldsymbol{\beta}_c\}}{\sum_{k=1}^{C}\exp\{\mathbf{x}_i^T\boldsymbol{\beta}_k\}}\right]^{y_{i,c}}$$

Log-likelihood is

$$\mathcal{LL}(\beta_0, B|Y, X) = \sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\left(\mathbf{x}_i^T\boldsymbol{\beta}_c - \log\left\{\sum_{k=1}^{C}\exp\{\mathbf{x}_i^T\boldsymbol{\beta}_k\}\right\}\right)$$

Multiclass Logistic Regression -- Regularized Log-Likelihood

Ridge regularized log-likelihood

$$\mathcal{PLL}(B|Y,X) = \sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\left(\mathbf{x}_i^T\boldsymbol{\beta}_c - \log\left\{\sum_{k=1}^{C}\exp\{\mathbf{x}_i^T\boldsymbol{\beta}_k\}\right\}\right) - \frac{\lambda}{2}\sum_{k=1}^{C}\sum_{j=1}^{p}\beta_{j,k}^2$$

Note that we keep the last column of B fixed at 0 to get rid of excess parameters

These parameters will not contribute to the regularization -- sum of their squares is 0

Cross-Entropy

Frequently you will encounter mentions of cross-entropy. It is negative log likelihood of multiclass logistic

$$\begin{aligned}\text{crossentropy}(B) &= -\mathcal{LL}(B|Y,X)\\ &= -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\left(\mathbf{x}_i^T\boldsymbol{\beta}_c - \log\left\{\sum_{k=1}^{C}\exp\{\mathbf{x}_i^T\boldsymbol{\beta}_k\}\right\}\right)\end{aligned}$$

Ridge regularized cross-entropy

$$\text{crossentropy}(B) = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\left(\mathbf{x}_i^T\boldsymbol{\beta}_c - \log\left\{\sum_{k=1}^{C}\exp\{\mathbf{x}_i^T\boldsymbol{\beta}_k\}\right\}\right) + \frac{\lambda}{2}\sum_{k=1}^{C}\sum_{j=1}^{p}\beta_{j,k}^2$$

Note the sign flip in the regularization

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:

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

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

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

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

On board we will work out

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

Hence

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

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

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

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

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

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

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

COMP 562 – Lecture 8

Multivariate Gaussian Distribution -- Dependent Case

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

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

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 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 we can obtain the conditional probability $p(y|\mathbf{x})$ using Bayes rule

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

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

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

This assumption **Conditional Independence of Features** underlies the **Naive Bayes** method

Naive Bayes

$$p(y = c|\pi) = \pi_c$$
$$p(\mathbf{x}|y = c, \theta) = \prod_j p(x_j|y = c, \theta_{j,c})$$

Parameters are

- π_c prior probability that a sample comes from the class c
- $\theta_{j,c}$ parameters for the j^{th} feature for class c

In general, there are many variants of Naive Bayes, you can choose different distributions for $p(x_j|y = c)$

- Gaussian -- continuous features
- Bernoulli -- binary features
- Binomial -- count of positive outcomes
- Categorical -- discrete features
- Multinomial -- count of particular discrete outcomes

Naive Bayes with Gaussian Features

We will assume that

$$x_j|y_c, \theta \sim \mathcal{N}\left(\theta_{j,c}, \sigma^2\right)$$

Each feature is Gaussian distributed around class specific mean and with shared spherical variance

Let's take a look at the data we generated earlier and read-off these parameters

joint Log-likelihood

$$\begin{aligned}
\mathcal{LL}(\theta, \pi|\mathbf{y}, X) &= \sum_i \log p(y_i, \mathbf{x}_i|\theta, \pi) && \text{definition of likelihood} \\
&= \sum_i \log p(y_i|\pi) + \log p(\mathbf{x}_i|y_i, \theta) && \text{factorization } p(y, \mathbf{x}) = p(y)p(\mathbf{x}|y) \\
&= \sum_i \log p(y_i|\pi) + \log \prod_j p(x_{j,i}|y_i, \theta_j) && \text{Naive Bayes assumption} \\
&= \sum_i \log p(y_i|\pi) + \sum_j \log p(x_{j,i}|y_i, \theta_j)
\end{aligned}$$

Note that we have not yet used our assumptions about distribution of $x_{j,i}$

Learning parameters for Naive Bayes with Gaussian features

joint Log-likelihood

$$\begin{aligned}
\mathcal{LL}(\theta, \pi|\mathbf{y}, X) &= \sum_i \left[\log p(y_i|\pi) + \sum_j \log p(x_{j,i}|y_i, \theta_j)\right] \\
&= \sum_i \left[\log \pi_{y_i} + \sum_j \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x_{j,i} - \theta_{j,y_i})^2\right\}\right] \\
&= \sum_i \left[\log \pi_{y_i} - \frac{1}{2}\sum_j \log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_j (x_{j,i} - \theta_{j,y_i})^2\right]
\end{aligned}$$

Note that parameters π_c and $\theta_{i,c}$ are only used for samples that belong to class c ($y_i = c$)

Hence, we can learn of parameters for each class separately

Learning Parameters for Naive Bayes with Gaussian Features

Closed form estimates for parameters are

$$\pi_c = \frac{\sum_i [y_i = c]}{\sum_i 1} \qquad \text{frequency of class } c \text{ in training data}$$

$$\theta_{j,c} = \frac{\sum_i [y_i = c] x_{i,j}}{\sum_i [y_i = c]} \qquad \text{average of feature } j \text{ among samples in class } c$$

$$\sigma = \frac{\sum_i (x_{j,i} - \theta_{j,y_i})^2}{\sum_i 1} \qquad \text{variance across all features}$$

Note $[x]$ is an indicator function, defined as

$$[x] = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Class Prediction using Naive Bayes with Gaussian Features

Recall that

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

After a little bit more manipulation

$$\log p(y = c|\mathbf{x}, \theta, \pi) = \log \pi_c - \sum_j \frac{1}{2\sigma^2}(x_{j,i} - \theta_{j,c})^2 + \text{const.}$$

Predicted class

$$y^* = \underset{c}{\operatorname{argmax}}\, \log \pi_c - \underbrace{\sum_j (x_{j,i} - \theta_{j,c})^2}_{\text{distance to class center}} + \text{const.}$$

- **Larger** K means **less bias** towards overestimating the true expected error (as training folds will be closer to the total dataset) but **typically higher variance and higher running time** (as you are getting closer to the limit case: Leave-One-Out CV)
- If cross-validation were averaging independent estimates, then with large K, one should see relatively lower variance between models; however, this is not true when training sets are highly correlated, which is what we typically deal with

Classification Performance -- Prediction Rate

Sensitivity, Recall, or True Positive Rate (TPR)

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = 1 - FNR$$

Specificity, Selectivity or True Negative Rate (TNR)

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} = 1 - FPR$$

False Positive Rate (FPR)

$$FPR = \frac{FP}{N} = \frac{FP}{TN + FP} = 1 - TNR$$

False Negative Rate (FNR)

$$FNR = \frac{FN}{P} = \frac{FN}{TP + FN} = 1 - TPR$$

These measures are computed from either +ve or -ve group, hence they don't depend on classes balance (prevalence = $\frac{P}{P+N}$)

Precision or Positive Predictive Value (PPV)

$$PPV = \frac{TP}{TP + FP}$$

Negative Predictive Value (NPV)

$$NPV = \frac{TN}{TN + FN}$$

Accuracy (ACC)

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + FN + TN + FP}$$

These measures are computed from both +ve and -ve groups, hence they depend on classes balance (prevalence = $\frac{P}{P+N}$)

Classification Performance -- ROC Curves

Predictions are based on a cutoff

$$p(y = 1|\mathbf{x}) > \tau$$

where τ is typically 0.5

This particular cutoff will result in a specific prediction rates; however, you may prefer to tradeoff false positives for false negatives -- health industry does

Classification Performance -- ROC Curves

 A good ROC curve – hugs top left corner: high TPR, low FPR

- A bad ROC curve – runs along diagonal: TPR equals the FPR

Probabilistic Interpretation of AUC

AUC Interpretation: AUC is the probability of correct ranking of a random "positive"-"negative" pair

- So, given a randomly chosen observation x_1 belonging to class 1, and a randomly chosen observation x belonging to class 0, the AUC is the probability that the evaluated classification algorithm will assign a higher score to x than to x_2, i.e., the conditional probability of $p(y = 1|x_1) > p(y = 1|x_2)$

AUC Computation: Among all "positive"-"negative" pairs in the dataset compute the proportion of those which are ranked correctly by the evaluated classification algorithm

$$\hat{AUC} = \frac{1}{P \times N} \sum_{i=1}^{P} \sum_{j=1}^{N} [p(y = 1|x_i) > p(y = 1|x_j)]$$