Lecture 2: Logistic Regression

STATS305B: Applied Statistics II

Scott Linderman

January 8, 2025

Recap

Last time...

- Basic distributions (Bernoulli, binomial, Poisson, categorical, multinomial)
- Basic inference (MLE, hypothesis tests, confidence intervals)
- Contingency tables (likelihood ratio test of independence)

Outline

Today, from contingency tables two logistic regression

- Setting up the model
- Parameter estimation
  - The hacky way: OLS
  - Maximum likelihood estimation
  - Gradient descent and Newton's method
  - Iteratively reweighted least squares (IRLS)
- Regularization
- Convergence rates

Contingency Tables with Binary Responses

A special case of contingency table analyses is when $X \in \{1, \ldots, I\}$ corresponds to a categorical feature (e.g., to which of I groups a data point belongs) and $Y \in \{0, 1\}$ is a binary response.

The corresponding tables are $I \times 2$. Normalizing each row yields a Bernoulli conditional distribution,

$$Y \mid X = i \sim \text{Bern}(\pi_{1|i}).$$

where, $\pi_{1|i} = \Pr(Y = 1 \mid X = i)$ or, equivalently, $\pi_{1|i} = \mathbb{E}[Y \mid X = i]$.

Conditional distributions are often the primary objects of interest.

One, Two, Many...

What if we have more than one feature, $X_1, \ldots, X_p$?

What if the features take on continuous values?

Idea: Model the conditional distribution directly. Let

- $Y \in \{0, 1\}$ denote a binary response
- $\boldsymbol{X} \in \mathbb{R}^p$ denote associated covariates.

E.g., Y could denote whether or not your favorite football team wins their match, and X could represent features of the match like whether its a home or away game, who their opponent is, etc.

One, Two, Many...

Model the conditional distribution as,

$$Y \mid \boldsymbol{X} = \boldsymbol{x} \sim \text{Bern}(\pi(\boldsymbol{x}))$$

where

$$\pi(\boldsymbol{x}) = \Pr(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}) = \mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}].$$

Standard regression setup.

Modeling choice: what functional form of $\pi(\boldsymbol{x})$?

Linear Regression

If you took STATS 305A, you know pretty much everything there is to know about linear regression with continuous response variables, $Y \in \mathbb{R}$. Why don't we just apply that same model to binary responses? Specifically, let,

$$\pi(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{x} = \sum_{j=1}^{p} \beta_j x_j.$$

Problem: linear model produces probabilities or expectations $\pi(\boldsymbol{x})$ outside $[0, 1]$

Logistic Regression

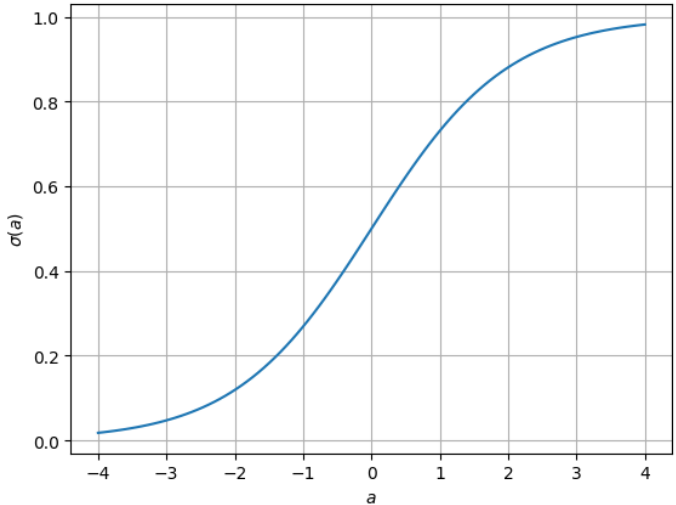
The idea is simple: keep the linear part of linear regression, but apply a **mean (aka inverse link) function**, $f : \mathbb{R} \mapsto [0, 1]$, to ensure $\pi(\boldsymbol{x})$ returns valid probabilities,

$$\pi(\boldsymbol{x}) = f(\boldsymbol{\beta}^\top \boldsymbol{x}).$$

There are infinitely many squashing nonlinearities that we could choose for f, but a particularly attractive choice is the **logistic (aka sigmoid) function**,

$$f(a) = \frac{e^a}{1 + e^a} = \frac{1}{1 + e^{-a}} \triangleq \sigma(a).$$

Logistic Function



Nice Properties of the Logistic Function

- Monotonically increasing
- Invertible: $\boldsymbol{\beta}^\top \boldsymbol{x}$ correspond to the **log odds** of the binary response since the inverse of the sigmoid function is the **logit function**,

$$\boldsymbol{\beta}^\top \boldsymbol{x} = \sigma^{-1}(\pi(\boldsymbol{x})) = \log \frac{\pi(\boldsymbol{x})}{1 - \pi(\boldsymbol{x})}.$$

- Simplifies some calculations for MLE

Another common mean function is the Gaussian CDF, and we'll consider that in a later chapter.

Hacky Parameter Estimation with OLS

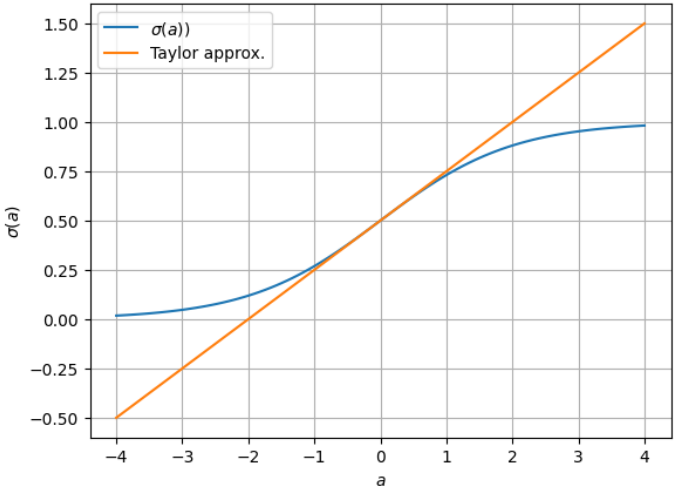
How do we estimate the parameters, $\hat{\boldsymbol{\beta}}$?

If the model were linear, then our first inclination might be to use ordinary least squares (OLS). Of course, the sigmoidal function above renders the model nonlinear.

Idea: What if we just used a Taylor approximation around the origin,

$$\sigma(a) \approx \sigma(0) + \sigma'(0)a = \frac{1}{2} + \frac{a}{4}$$

Hacky Parameter Estimation with OLS



Hacky Parameter Estimation with OLS

Then under this model,

$$\mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] \approx \frac{1}{2} + \frac{\boldsymbol{\beta}^\top \boldsymbol{x}}{4}$$

or equivalently,

$$\mathbb{E}[Z \mid \boldsymbol{X} = \boldsymbol{x}] \approx \boldsymbol{\beta}^\top \boldsymbol{x}$$

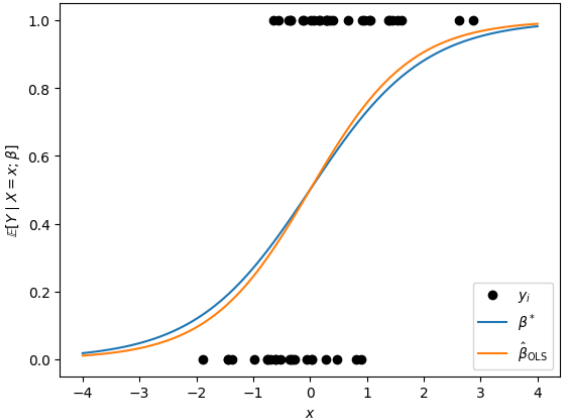
where $Z = 4(Y - \frac{1}{2}) \in \{-2, +2\}$ is an adjusted response variable.

Given a set of n observations of features and (adjusted) responses, $\{\boldsymbol{x}_i, z_i\}_{i=1}^n$, the OLS estimate is,

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = \left(\sum_{i=1}^n \boldsymbol{x}_i \boldsymbol{x}_i^\top\right)^{-1} \left(\sum_{i=1}^n \boldsymbol{x}_i z_i\right)$$

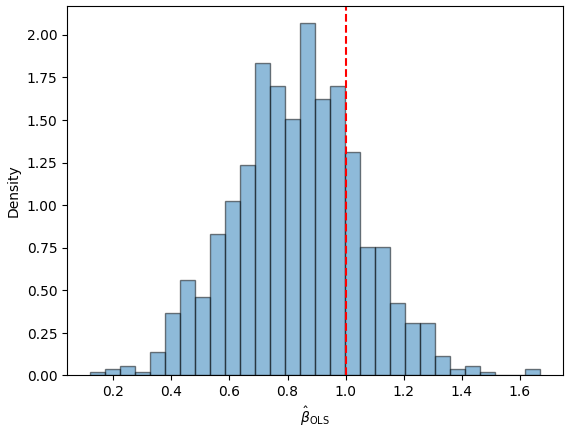
Demo

Simulated dataset with scalar covariates $X_i \sim \mathrm{N}(0, 1)$ and responses $Y_i \mid X_i = x_i \sim \mathrm{Bern}(\sigma(\beta^* x_i))$ for $\beta^* = 1$.



Demo

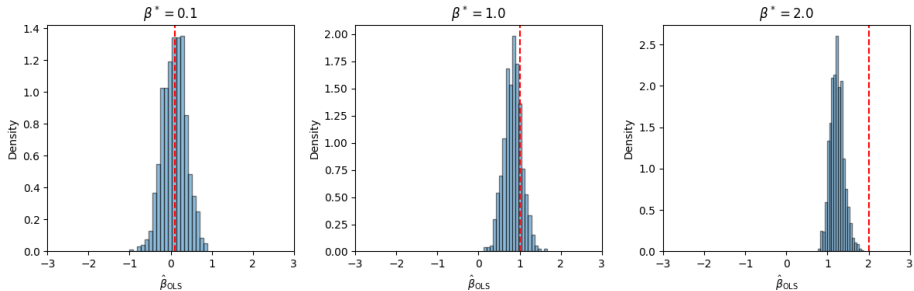
Not too shabby! Let's try it with a bunch of simulated datasets.



Demo

It looks a bit biased, but it's not terrible.

What if we try for other values of β^*?



Question: Why do you think the OLS estimator becomes more biased as β^* grows?

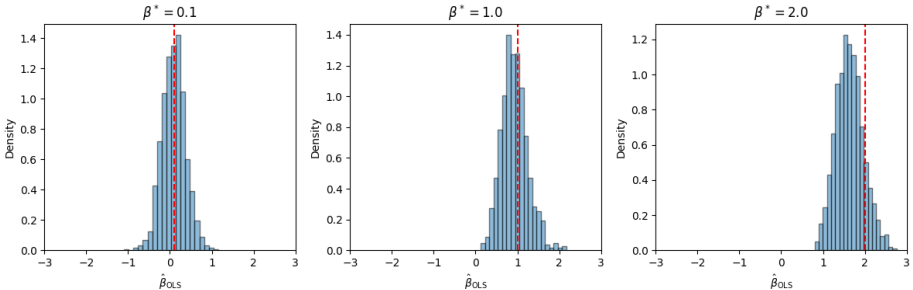
Maximum Likelihood Estimation

For standard linear regression with independent Gaussian responses and homoskedastic noise, $\hat{\boldsymbol{\beta}}_{\text{MLE}} = \hat{\boldsymbol{\beta}}_{\text{OLS}}$.

Unfortunately, for logistic regression, there isn't a closed form for $\hat{\boldsymbol{\beta}}_{\text{MLE}}$. However, we can use standard optimization techniques to compute it.

There are plenty of standard implementations of maximum likelihood estimation for logistic regression models. Let's see how scikit-learn's implementation fares on the simulated datasets above.

Demo



Deriving the MLE

Maximizing the likelihood is equivalent to minimizing the (average) negative log likelihood for a collection of covariates and responses, $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$,

$$
\begin{aligned}
\mathscr{L}(\boldsymbol{\beta}) &= -\frac{1}{n}\sum_{i=1}^{n} \log \text{Bern}(y_i; \pi(\boldsymbol{x}_i)) \\
&= -\frac{1}{n}\sum_{i=1}^{n} y_i \log \pi(\boldsymbol{x}_i) + (1-y_i)\log(1-\pi(\boldsymbol{x}_i)) \\
&= -\frac{1}{n}\sum_{i=1}^{n} y_i \log \frac{\pi(\boldsymbol{x}_i)}{1-\pi(\boldsymbol{x}_i)} + \log(1-\pi(\boldsymbol{x}_i)) \\
&= -\frac{1}{n}\sum_{i=1}^{n} y_i \boldsymbol{\beta}^\top \boldsymbol{x}_i + \log(1-\pi(\boldsymbol{x}_i)).
\end{aligned}
$$

Since $\boldsymbol{\beta}^\top \boldsymbol{x}_i$ is the log odds, as shown above.

Deriving the MLE

Plugging in the definition of $\pi(\boldsymbol{x}_i)$, the second term simplifies as well,

$$\begin{aligned}\mathscr{L}(\boldsymbol{\beta}) &= -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\boldsymbol{\beta}^\top\boldsymbol{x}_i + \log\left(1 - \frac{e^{\boldsymbol{\beta}^\top\boldsymbol{x}_i}}{1+e^{\boldsymbol{\beta}^\top\boldsymbol{x}_i}}\right)\right] \\ &= -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\boldsymbol{\beta}^\top\boldsymbol{x}_i - \log\left(1+e^{\boldsymbol{\beta}^\top\boldsymbol{x}_i}\right)\right].\end{aligned}$$

Computing the Gradient

To minimize the negative log likelihood, take the gradient,

$$\begin{aligned}\nabla \mathscr{L}(\boldsymbol{\beta}) &= -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \boldsymbol{x}_i - \frac{e^{\boldsymbol{\beta}^\top \boldsymbol{x}_i}}{1+e^{\boldsymbol{\beta}^\top \boldsymbol{x}_i}}\boldsymbol{x}_i\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sigma(\boldsymbol{\beta}^\top \boldsymbol{x}_i)\right)\boldsymbol{x}_i.\end{aligned}$$

The gradient is a weighted sum of the covariates, and the weights are the residuals $y_i - \sigma(\boldsymbol{x}_i^\top \boldsymbol{\beta})$, i.e., the difference between the observed and expected response.

Intuition: move in the direction of covariates where the residual is positive (we are underestimating the mean), and opposite covariates where the residual is negative (where we are overestimating the mean).

Computing the Gradient

Algorithm:

$$\boldsymbol{\beta}_{t+1} \leftarrow \boldsymbol{\beta}_t - \alpha_t \nabla \mathscr{L}(\boldsymbol{\beta}_t),$$

where $\alpha_t \in \mathbb{R}_+$ is the step-size at iteration *i* of the algorithm.

If the step sizes are chosen appropriately and the objective is well behaved, the alorithm converges to at least a local optimum of the log likelihood.

Convexity of the Log Likelihood

If the objective is convex, then all local optima are also global optima, and we can give stronger guarantees on gradient descent. To check the convexity of the log likelihood, we need to compute its Hessian,

$$\nabla^2 \mathscr{L}(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \sigma'(\boldsymbol{\beta}^\top \boldsymbol{x}_i) \boldsymbol{x}_i \boldsymbol{x}_i^\top$$

where $\sigma'(a)$ is the derivative of the logistic function. That is,

$$\begin{aligned} \sigma'(a) &= \frac{\mathrm{d}}{\mathrm{d}a} \sigma(a) \\ &= \frac{e^a}{(1+e^a)^2} \\ &= \sigma(a)(1-\sigma(a)). \end{aligned}$$

Convexity of the Log Likelihood

Plugging this in,

$$\begin{aligned}\nabla^2 \mathscr{L}(\boldsymbol{\beta}) &= \frac{1}{n}\sum_{i=1}^{n} \sigma(\boldsymbol{\beta}^\top \boldsymbol{x}_i)(1-\sigma(\boldsymbol{\beta}^\top \boldsymbol{x}_i))\boldsymbol{x}_i \boldsymbol{x}_i^\top \\ &= \frac{1}{n}\sum_{i=1}^{n} w_i \boldsymbol{x}_i \boldsymbol{x}_i^\top,\end{aligned}$$

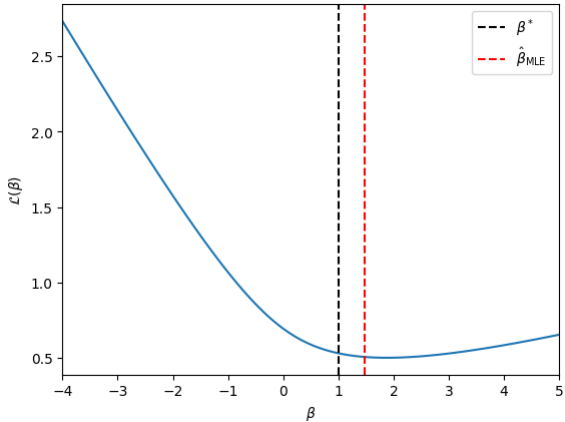
where the weights are, $w_i = \sigma(\boldsymbol{\beta}^\top \boldsymbol{x}_i)(1-\sigma(\boldsymbol{\beta}^\top \boldsymbol{x}_i)) = \text{Var}[Y \mid \boldsymbol{X} = \boldsymbol{x}_i]$.

The Hessian is a weighted sum of outer products of covariates where the weights are equal to the conditional variance, which are non-negative.

Since $w_i > 0$, the Hessian is **positive semi-definite**, which implies that the negative log likelihood is convex.

Example

As an example, let's plot the negative log likelihood for a scalar covariate example, as above.



Pathologies in the Separable Regime

Suppose the two classes are *linearly separable*,

$$\exists \boldsymbol{u} \in \mathbb{S}_{p-1} \text{ s.t. } \begin{cases} \boldsymbol{u}^\top \boldsymbol{x}_i > 0 & \text{if } y_i = 1 \\ \boldsymbol{u}^\top \boldsymbol{x}_i < 0 & \text{if } y_i = 0 \end{cases}$$

Now let $\boldsymbol{\beta} = c\boldsymbol{u}$ for any $c \in \mathbb{R}_+$. We have

$$\lim_{c \to \infty} \sigma(c\boldsymbol{u}^\top \boldsymbol{x}_i) = y_i.$$

In this limit, the model is saturated: $\Pr(y_i \mid \boldsymbol{x}_i) = 1$ for all $i = 1, \ldots, n$; the negative log likelihood goes to $\lim_{c \to \infty} \mathscr{L}(c\boldsymbol{u}) = 0$; and the MLE does not exist since $\boldsymbol{\beta}^\star$ diverges.

If we run gradient descent in this setting, the magnitude of the estimate $\hat{\boldsymbol{\beta}}$ will grow without bound, while the objective converges to zero.

L_2-Regularization

These pathologies can be averted with a little regularization,

$$\mathscr{L}(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n} \log \text{Bern}(y_i; \sigma(\boldsymbol{\beta}^\top \boldsymbol{x}_i)) + \frac{\gamma}{2}\|\boldsymbol{\beta}\|_2^2.$$

The regularizer penalizes larger values of the weights, $\boldsymbol{\beta}$, and the **hyperparameter** $\gamma \in \mathbb{R}_+$ sets the strength of the penalty. Even in the linearly separable regime, the maximizer is finite.

Now, the gradient and Hessian are,

$$\nabla \mathscr{L}(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n} \big(y_i - \sigma(\boldsymbol{x}_i^\top \boldsymbol{\beta})\big)\boldsymbol{x}_i + \gamma \boldsymbol{\beta}$$

$$\nabla^2 \mathscr{L}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} w_i \boldsymbol{x}_i \boldsymbol{x}_i^\top + \gamma \boldsymbol{I}.$$

Choosing the Hyperparameter

There are many ways to select a value of γ.

One approach is to *not* choose a single value and instead try to compute the **regularization path**; i.e., the solution $\hat{\boldsymbol{\beta}}(\gamma)$ for a range of $\gamma \in [0, \infty)$.

Another is to hold out a fraction of data and use cross-validation to select the hyperparameter setting that yields the best performance on the held-out data.

Bayesian Perspective

From a Bayesian perspective, we can think of the regularizer as a **prior log probability**.

Here, the regularizer corresponds to a spherical Gaussian prior,

$$\boldsymbol{\beta} \sim \mathrm{N}(\mathbf{0}, (\gamma n)^{-1}\boldsymbol{I}),$$

where **precision (inverse covariance)** $\gamma n \boldsymbol{I}$.

Minimizing the objective above corresponds to doing **maximum a posteriori (MAP)** estimation in the Bayesian model.

We'll talk more about Bayesian methods in the coming weeks.

Newton's Method

Gradient descent leverages the gradient at $\boldsymbol{\beta}$ to determine the update. Newton's method uses the Hessian to inform the update as well, and in doing so it can achieve considerably faster convergence.

The second order Taylor approximation of $\mathscr{L}$ around $\boldsymbol{\beta}$ is

$$\mathscr{L}(\boldsymbol{\beta}') \approx \mathscr{L}(\boldsymbol{\beta}) + \nabla\mathscr{L}(\boldsymbol{\beta})^\top(\boldsymbol{\beta}' - \boldsymbol{\beta}) + \frac{1}{2}(\boldsymbol{\beta}' - \boldsymbol{\beta})^\top\nabla^2\mathscr{L}(\boldsymbol{\beta})(\boldsymbol{\beta}' - \boldsymbol{\beta}) \triangleq \widetilde{\mathscr{L}}(\boldsymbol{\beta}')$$

The stationary point of $\widetilde{\mathscr{L}}(\boldsymbol{\beta}')$ is at

$$\begin{aligned} \nabla\widetilde{\mathscr{L}}(\boldsymbol{\beta}') &= \nabla\mathscr{L}(\boldsymbol{\beta}) + \nabla^2\mathscr{L}(\boldsymbol{\beta})(\boldsymbol{\beta}' - \boldsymbol{\beta}) = 0 \\ \implies \boldsymbol{\beta}' &= \boldsymbol{\beta} - [\nabla^2\mathscr{L}(\boldsymbol{\beta})]^{-1}\nabla\mathscr{L}(\boldsymbol{\beta}), \end{aligned}$$

assuming the Hessian is invertible. When $\nabla^2\mathscr{L}(\boldsymbol{\beta}) \succ 0$ — i.e., when the Hessian is positive definite — the inverse Hessian exists and the stationary point is a minimizer.

Newton's Method

Vanilla Newton's method applies this update repeatedly until convergence, forming a quadratic approximation and then minimizing it.

Damped Newton's method adds a step size $\alpha_t < 1$ to improve stability,

$$\boldsymbol{\beta}_{t+1} \leftarrow \boldsymbol{\beta}_t - \alpha_t [\nabla^2 \mathscr{L}(\boldsymbol{\beta}_t)]^{-1} \nabla \mathscr{L}(\boldsymbol{\beta}_t),$$

The step size can be chosen by backtracking line search.

Question: Compare the Newton update to that of gradient descent. How do they differ?

Iteratively Reweighted Least Squares

Newton's method can be viewed as iteratively solving a weighted least squares problem.

Write the gradient and Hessian in matrix form,

$$\nabla \mathscr{L}(\boldsymbol{\beta}_t) = -\frac{1}{n}\boldsymbol{X}^\top(\boldsymbol{y} - \hat{\boldsymbol{y}}_t)$$
$$\nabla^2 \mathscr{L}(\boldsymbol{\beta}_t) = \frac{1}{n}\boldsymbol{X}^\top \boldsymbol{W}_t \boldsymbol{X}$$

where

- $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ is the design matrix with rows $\boldsymbol{x}_i^\top$
- $\boldsymbol{y} \in \{0, 1\}^n$ is the vector of binary responses
- $\hat{\boldsymbol{y}}_t = \sigma(\boldsymbol{X}\boldsymbol{\beta}_t) \in [0, 1]^n$ is the vector of predicted response means
- $\boldsymbol{W}_t = \text{diag}([w_{t,1}, \ldots, w_{t,n}])$ with $w_{t,i} = \sigma(\boldsymbol{x}_i^\top \boldsymbol{\beta}_t)(1 - \sigma(\boldsymbol{x}_i^\top \boldsymbol{\beta}_t))$ is the diagonal weight matrix.

Iteratively Reweighted Least Squares

Then the (undamped) Newton update is,

$$\begin{aligned}\boldsymbol{\beta}_{t+1} &= \boldsymbol{\beta}_t - [\nabla^2 \mathscr{L}(\boldsymbol{\beta}_t)]^{-1} \nabla \mathscr{L}(\boldsymbol{\beta}_t) \\ &= \boldsymbol{\beta}_t + [\boldsymbol{X}^\top \boldsymbol{W}_t \boldsymbol{X}]^{-1} \boldsymbol{X}^\top (\boldsymbol{y} - \hat{\boldsymbol{y}}_t) \\ &= [\boldsymbol{X}^\top \boldsymbol{W}_t \boldsymbol{X}]^{-1} \boldsymbol{X}^\top \boldsymbol{W}_t \boldsymbol{X} \boldsymbol{\beta}_t + [\boldsymbol{X}^\top \boldsymbol{W}_t \boldsymbol{X}]^{-1} \boldsymbol{X}^\top (\boldsymbol{y} - \hat{\boldsymbol{y}}_t) \\ &= [\boldsymbol{X}^\top \boldsymbol{W}_t \boldsymbol{X}]^{-1} \boldsymbol{X}^\top \boldsymbol{W}_t \boldsymbol{z}_t\end{aligned}$$

where

$$\boldsymbol{z}_t = \boldsymbol{X}\boldsymbol{\beta}_t + \boldsymbol{W}_t^{-1}(\boldsymbol{y} - \hat{\boldsymbol{y}}_t).$$

This is a **weighted least squares** problem with **weights** $w_{t,i}$ and **adjusted (or working) responses** $z_{t,i}$.

Iteratively Reweighted Least Squares

How can we interpret the working responses? We can view them as the real responses mapped through a Taylor approximation of the link (inverse mean) function,

$$\begin{aligned}\sigma^{-1}(y_i) &\approx \sigma^{-1}(\hat{y}_{t,i}) + (y_i - \hat{y}_{t,i})[\sigma^{-1}]'(\hat{y}_{t,i}) \\ &= \boldsymbol{x}_i^\top \boldsymbol{\beta}_t + \frac{(y_i - \hat{y}_{t,i})}{w_{t,i}} \\ &\triangleq z_{t,i}.\end{aligned}$$

Asymptotic Covariance of MLE

What other insight can we glean from the Hessian? Recall our discussion of the asymptotic normality of the MLE from Lecture 1. For iid observations $Y_i \overset{\text{iid}}{\sim} p(\cdot; \boldsymbol{\theta})$ for $i = 1, \ldots, n$, the asymptotic covariance is $\mathscr{I}(\boldsymbol{\theta})^{-1}/n$, where

$$\mathscr{I}(\boldsymbol{\theta}) = -\mathbb{E}[\nabla^2_{\boldsymbol{\theta}} \log p(Y; \boldsymbol{\theta})]$$

is the Fisher information matrix.

For logistic regression, we have n independent but *not* identically distributed observations. In this case, the asymptotic covariance follows the same form. It is given by the inverse of the Fisher information matrix; i.e., the inverse of the negative expected Hessian of the log likelihood,

$$\mathscr{I}(\boldsymbol{\beta}) = -\sum_{i=1}^{N} \mathbb{E}[\nabla^2_{\boldsymbol{\beta}} \log p(Y_i \mid \boldsymbol{X}_i = \boldsymbol{x}_i; \boldsymbol{\beta})].$$

(Note that this includes the iid formula as a special case.)

Asymptotic Covariance of MLE

Substituting the form of the Hessian from above,

$$\begin{aligned}\mathscr{I}(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \mathbb{E}[\mathrm{Var}[Y_i \mid \boldsymbol{X} = \boldsymbol{x}_i]\boldsymbol{x}_i\boldsymbol{x}_i^\top] \\ &= \sum_{i=1}^{n} w_i \boldsymbol{x}_i \boldsymbol{x}_i^\top\end{aligned}$$

where $w_i = \sigma(\boldsymbol{\beta}^\top \boldsymbol{x}_i)(1-\sigma(\boldsymbol{\beta}^\top \boldsymbol{x}_i))$. Finally, we evaluate the Fisher information the MLE to obtain the asymptotic covariance estimate,

$$\widehat{\mathrm{Cov}}(\hat{\boldsymbol{\beta}}) = [\mathscr{I}(\hat{\boldsymbol{\beta}})]^{-1}.$$

Like before, we can use the asymptotic covariance estimate to derive Wald confidence intervals for the parameters and perform hypothesis tests.

Asymptotic Covariance of MLE

Caution: Remember, Wald confidence intervals are only as good as the asymptotic normality assumption. When the likelihod is not well approximated by a quadratic, the covariance estimate will be poor, and the confidence intervals will be invalid. When might the Gaussian approximation not hold? :::

Converge Rate of Gradient Descent

To determine when and at what rate gradient descent converges, we need to know more about the eigenvalues of the Hessian.

If we can bound the maximum eigenvalue of the Hessian by L, then we can obtain a quadratic upper bound on the negative log likelihood,

$$\mathscr{L}(\boldsymbol{\beta}') \leq \mathscr{L}(\boldsymbol{\beta}) + \nabla \mathscr{L}(\boldsymbol{\beta})^\top (\boldsymbol{\beta}' - \boldsymbol{\beta}) + \frac{L}{2}(\boldsymbol{\beta}' - \boldsymbol{\beta})^\top (\boldsymbol{\beta}' - \boldsymbol{\beta}).$$

That means the negative log likelihood is an L-smooth function.

Converge Rate of Gradient Descent

Example: Bounded Covariates

If the covariates have bounded norm, $||\boldsymbol{x}_i||_2 \leq B$, then we can bound the maximum eigenvalue of the Hessian by,

$$\begin{aligned}
\lambda_{\max} &= \max_{\boldsymbol{u} \in \mathbb{S}_{p-1}} \boldsymbol{u}^\top \nabla^2 \mathscr{L}(\boldsymbol{\beta}) \boldsymbol{u} \\
&= \max_{\boldsymbol{u} \in \mathbb{S}_{p-1}} \frac{1}{n} \sum_{i=1}^{n} w_i \boldsymbol{u}^\top \boldsymbol{x}_i \boldsymbol{x}_i^\top \boldsymbol{u} \\
&\leq \frac{B^2}{4}
\end{aligned}$$

since the weights are the conditional variances of Bernoulli random variables, which are at most $\frac{1}{4}$, and since $\boldsymbol{u}^\top \boldsymbol{x}_i \leq B$ for all unit vectors $\boldsymbol{u} \in \mathbb{S}_{p-1}$ (the unit sphere embedded in $\mathbb{R}^p$). This isn't meant to be a tight upper bound.

Converge Rate of Gradient Descent

If we run gradient descent with a constant step size of $\alpha = 1/L$, then the algorithm converges at a rate of $1/t$, which means that after t iterations

$$\mathscr{L}(\boldsymbol{\beta}^\star) - \mathscr{L}(\boldsymbol{\beta}_t) \leq \frac{L}{t} \|\boldsymbol{\beta}^\star - \boldsymbol{\beta}_0\|_2^2,$$

where $\boldsymbol{\beta}_0$ is the initial setting of the parameters and $\boldsymbol{\beta}^\star$ is the global optimum.

Put differently, if we want a gap of at most epsilon, we need to run $t \sim \mathcal{O}(1/\epsilon)$ iterations of gradient descent. Put differently, if we want to reduce ϵ by a factor of 100, we need to run around 100 times as many iterations. This is called a **sub-linear convergence** rate.

Converge Rate of Gradient Descent with Regularization

With regularization, we can also **lower bound** the objective by a quadratic function,

$$\mathscr{L}(\boldsymbol{\beta}') \geq \mathscr{L}(\boldsymbol{\beta}) + \nabla\mathscr{L}(\boldsymbol{\beta})^\top(\boldsymbol{\beta}' - \boldsymbol{\beta}) + \frac{\mu}{2}(\boldsymbol{\beta}' - \boldsymbol{\beta})^\top(\boldsymbol{\beta}' - \boldsymbol{\beta})$$

for $\mu > 0$, which means the objective is μ-**strongly convex**.

For twice differentiable objectives, the minimum eigenvalue of Hessian provides such a lower bound, $\mu = \lambda_{\min}$. In particular, we know that minimum eigenvalue of $\nabla^2\mathscr{L}(\boldsymbol{\beta})$ is at least γ. (This bound is achieved when the data are linearly separable, the model is saturated, and the conditional variances are all zero.)

For a L-smooth and μ-strongly convex function with stepsize $\alpha = 1/L$, gradient descent has the following convergence guarantee,

$$\mathscr{L}(\boldsymbol{\beta}_{t+1}) - \mathscr{L}(\boldsymbol{\beta}^\star) \leq \left(1 - \frac{\mu}{L}\right)(\mathscr{L}(\boldsymbol{\beta}_t) - \mathscr{L}(\boldsymbol{\beta}^\star)).$$

Converge Rate of Gradient Descent with Regularization

Applying this bound recursively yields that,

$$\mathscr{L}(\boldsymbol{\beta}_t) - \mathscr{L}(\boldsymbol{\beta}^\star) \leq \left(1 - \frac{\mu}{L}\right)^t (\mathscr{L}(\boldsymbol{\beta}_0) - \mathscr{L}(\boldsymbol{\beta}^\star)).$$

If we want to find the number of iterations t to bound the gap by at most ϵ, we need to solve for t in

$$\left(1 - \frac{\mu}{L}\right)^t (\mathscr{L}(\beta_0) - \mathscr{L}(\beta^*)) \leq \epsilon.$$

This inequality is equivalent to taking the log of both sides

$$\log(\mathscr{L}(\beta_0) - \mathscr{L}(\beta^*)) + t \log\left(1 - \frac{\mu}{L}\right) \leq \log \epsilon.$$

Converge Rate of Gradient Descent with Regularization

We can further upper bound the LHS by using the inequality $\log(1-x) \leq -x$ to get

$$\log(\mathscr{L}(\beta_0) - \mathscr{L}(\beta^*)) - \frac{\mu t}{L} \leq \log \epsilon. \tag{1}$$

So, we need to run $t \geq \frac{L}{\mu} \log \frac{\mathscr{L}(\beta_0) - \mathscr{L}(\beta^*)}{\epsilon} \sim \log \frac{1}{\epsilon}$ iterations of gradient descent. If we want to reduce ϵ by a factor of 100, we only need to run around $\log 100$ times as many iterations. This is called **linear convergence**.

Converge Rate of Newton's Method

Under certain conditions — if the objective is strongly convex, the Hessian is Lipschitz continuous, and we start near enough to the global optimum — Newton's method achieves **second order convergence**, meaning

$$\|\boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}^\star\|_2 \leq (c\|\boldsymbol{\beta}_t - \boldsymbol{\beta}^\star\|_2)^2$$

for some positive constant c, provided we start with $\boldsymbol{\beta}_0$ close enough to $\boldsymbol{\beta}^\star$. Applying this bound recursively yields that,

$$\|\boldsymbol{\beta}_t - \boldsymbol{\beta}^\star\|_2 \leq (c\|\boldsymbol{\beta}_0 - \boldsymbol{\beta}^\star\|_2)^{2^t}.$$

Put differently, if we start with $\|\boldsymbol{\beta}_0 - \boldsymbol{\beta}^\star\| < c^{-1}$, then we need $t \sim \mathcal{O}(\log\log\frac{1}{\epsilon})$ iterations to obtain an error of ϵ. Since the double log grows incredibly slowly, this statement says that we effectively need a constant number of iterations for Newton's method to converge in this regime.

For more information on convergence rates of gradient descent and Newton's method, see {cite:t}`boyd2004convex`, ch. 9.

Conclusion

One, two, many… with logistic regression, we can begin modeling relationships between binary response variables and (possibly arbitrary-valued) covariates. Next, we'll see how to expand from logistic regression to generalized linear models for exponential family responses.