AM 230 - Course Project I
Binary Outcome Prediction

Luke Catalano
UC Santa Cruz, M.S. Quantitative Economics

January 2026

Problem 2.1.

For a single (x^i, y^i), define the per-sample logistic loss:

$$l^i(\theta) = log(1 + e^{s_\theta(x^i)}) - y^i(s_\theta(x^i))$$

Where $s_\theta(x^i) = w_1 x_1^i + w_2 x_2^i + b$.

a) Compute the gradient of the per-sample logistic loss with respect to the parameters:

$$\nabla_\theta l^i(\theta) = \begin{bmatrix} \frac{\partial l^i}{\delta w_1} \\ \frac{\partial l^i}{\delta w_2} \\ \frac{\partial l^i}{\delta b} \end{bmatrix}$$

For the first term, through the chain rule we can derive:

$$\frac{\partial l^i}{\partial w_1} = \frac{1}{1 + e^{s_\theta(x^i)}} \cdot e^{s_\theta(x^i)} \cdot \frac{\partial s_\theta(x^i)}{\partial w_1} - y^i \cdot \frac{\partial s_\theta(x^i)}{\partial w_1}$$

Note that $\frac{\partial s_\theta(x^i)}{\partial w_1}$ simplifies to x_1^i, so we can factor it out:

$$\frac{\partial l^i}{\partial w_1} = \frac{e^{s_\theta(x^i)}}{1 + e^{s_\theta(x^i)}} \cdot x_1^i - y^i \cdot x_1^i = x_1^i \left(\frac{e^{s_\theta(x^i)}}{1 + e^{s_\theta(x^i)}} - y^i \right)$$

The same follows for the second term:

$$\frac{\partial l^i}{\partial w_2} = \frac{1}{1 + e^{s_\theta(x^i)}} \cdot e^{s_\theta(x^i)} \cdot \frac{\partial s_\theta(x^i)}{\partial w_2} - y^i \cdot \frac{\partial s_\theta(x^i)}{\partial w_2}$$

Following the same simplification and factoring:

$$\frac{\partial l^i}{\partial w_2} = \frac{e^{s_\theta(x^i)}}{1 + e^{s_\theta(x^i)}} \cdot x_2^i - y^i \cdot x_2^i = x_2^i \left(\frac{e^{s_\theta(x^i)}}{1 + e^{s_\theta(x^i)}} - y^i \right)$$

Next, we solve for the partial derivative with respect to the intercept parameter:

$$\frac{\partial l^i}{\partial b} = \frac{1}{1 + e^{s_\theta(x^i)}} \cdot e^{s_\theta(x^i)} - y^i \cdot \frac{\partial s_\theta(x^i)}{\partial b} = \frac{e^{s_\theta(x^i)}}{1 + e^{s_\theta(x^i)}} - y^i$$

We define the sigmoid probability term as p^i, and define the gradient of the per-sample logistic loss

$$\sigma(s)^i = \frac{e^{s_\theta(x^i)}}{1 + e^{s_\theta(x^i)}} \implies \nabla_\theta l^i(\theta) = \begin{bmatrix} x_1^i(\sigma(s)^i - y^i) \\ x_2^i(\sigma(s)^i - y^i) \\ \sigma(s)^i - y^i \end{bmatrix}$$

b) Find the gradient of the total loss function:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\Big[log(1+e^{s_\theta(x^i)}) - y^i(s_\theta(x^i))\Big]$$

We define the gradient of the total loss function as:

$$\nabla_\theta L(\theta) = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \frac{\partial L}{\partial b} \end{bmatrix}$$

We take the partial derivative of the total loss function with respect to each model parameter:

$$\frac{\partial L}{\partial w_1} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{1+e^{s_\theta(x^i)}}\cdot e^{s_\theta(x^i)}\cdot\frac{\partial s_\theta(x^i)}{\partial w_1} - y^i\cdot\frac{\partial s_\theta(x^i)}{\partial w_1}\right] \implies \frac{1}{N}\sum_{i=1}^{N}\left[\frac{e^{s_\theta(x^i)}}{1+e^{s_\theta(x^i)}}\cdot x_1^i - y^i\cdot x_1^i\right]$$

$$\frac{\partial L}{\partial w_2} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{1+e^{s_\theta(x^i)}}\cdot e^{s_\theta(x^i)}\cdot\frac{\partial s_\theta(x^i)}{\partial w_2} - y^i\cdot\frac{\partial s_\theta(x^i)}{\partial w_2}\right] \implies \frac{1}{N}\sum_{i=1}^{N}\left[\frac{e^{s_\theta(x^i)}}{1+e^{s_\theta(x^i)}}\cdot x_2^i - y^i\cdot x_2^i\right]$$

$$\frac{\partial L}{\partial b} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{1+e^{s_\theta(x^i)}}\cdot e^{s_\theta(x^i)}\cdot\frac{\partial s_\theta(x^i)}{\partial b} - y^i\cdot\frac{\partial s_\theta(x^i)}{\partial b}\right] \implies \frac{1}{N}\sum_{i=1}^{N}\left[\frac{e^{s_\theta(x^i)}}{1+e^{s_\theta(x^i)}} - y^i\right]$$

We can simplify this expression similarly to our derivation of the gradient of the per-sample loss function:

$$\sigma(s) = \frac{1}{1+e^{-s}} = \frac{e^s}{1+e^s} \implies \nabla_\theta L(\theta) = \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N} x_1^i(\sigma(s_\theta(x^i)) - y^i) \\ \frac{1}{N}\sum_{i=1}^{N} x_2^i(\sigma(s_\theta(x^i)) - y^i) \\ \frac{1}{N}\sum_{i=1}^{N}(\sigma(s_\theta(x^i)) - y^i) \end{bmatrix}$$

Problem 2.2.

(a) Show that the Hessian matrix of the per-sample logistic loss is

$$\nabla_\theta^2 \ell^i(\theta) = \sigma'(s_\theta(x^i)) \begin{bmatrix} x_1^i \\ x_2^i \\ 1 \end{bmatrix} \begin{bmatrix} x_1^i & x_2^i & 1 \end{bmatrix}$$

Where the derivative of the sigmoid function is

$$\sigma'(s) = \sigma(s)(1-\sigma(s))$$

First, we set up the equality:

$$\nabla_\theta^2 \ell^i(\theta) = \begin{bmatrix} \frac{\partial\ell^2}{\partial^2 w_1^2} & \frac{\partial\ell^2}{\partial^2 w_1 w_2} & \frac{\partial\ell^2}{\partial^2 w_1 b} \\ \frac{\partial\ell^2}{\partial^2 w_1 w_2} & \frac{\partial\ell^2}{\partial^2 w_2^2} & \frac{\partial\ell^2}{\partial^2 w_2 b} \\ \frac{\partial\ell^2}{\partial^2 w_1 b} & \frac{\partial\ell^2}{\partial^2 w_2 b} & \frac{\partial\ell^2}{\partial^2 b^2} \end{bmatrix} = [\sigma(s)(1-\sigma(s))] \begin{bmatrix} x_1^i \\ x_2^i \\ 1 \end{bmatrix} \begin{bmatrix} x_1^i & x_2^i & 1 \end{bmatrix}$$

Simplifying the **LHS**:

$$\sigma(s)' \begin{bmatrix} x_1^i \\ x_2^i \\ 1 \end{bmatrix} \begin{bmatrix} x_1^i & x_2^i & 1 \end{bmatrix} = \sigma(s)' \begin{bmatrix} (x_1^i)^2 & x_1^i x_2^i & x_1^i \\ x_1^i x_2^i & (x_2^i)^2 & x_2^i \\ x_1^i & x_2^i & 1 \end{bmatrix} = \begin{bmatrix} \sigma(s)'(x_1^i)^2 & \sigma(s)' x_1^i x_2^i & \sigma(s)' x_1^i \\ \sigma(s)' x_1^i x_2^i & \sigma(s)'(x_2^i)^2 & \sigma(s)' x_2^i \\ \sigma(s)' x_1^i & \sigma(s)' x_2^i & \sigma(s)' \end{bmatrix}$$

This leaves us with the simplified equality of two symmetric matrices, a key step is showing they are equivalent. We now only need to solve 6 partial derivatives in the upper triangle:

$$\begin{bmatrix} \frac{\partial\ell^2}{\partial^2 w_1^2} & \frac{\partial\ell^2}{\partial^2 w_1 w_2} & \frac{\partial\ell^2}{\partial^2 w_1 b} \\ \frac{\partial\ell^2}{\partial^2 w_1 w_2} & \frac{\partial\ell^2}{\partial^2 w_2^2} & \frac{\partial\ell^2}{\partial^2 w_2 b} \\ \frac{\partial\ell^2}{\partial^2 w_1 b} & \frac{\partial\ell^2}{\partial^2 w_2 b} & \frac{\partial\ell^2}{\partial^2 b^2} \end{bmatrix} = \begin{bmatrix} \sigma(s)'(x_1^i)^2 & \sigma(s)' x_1^i x_2^i & \sigma(s)' x_1^i \\ \sigma(s)' x_1^i x_2^i & \sigma(s)'(x_2^i)^2 & \sigma(s)' x_2^i \\ \sigma(s)' x_1^i & \sigma(s)' x_2^i & \sigma(s)' \end{bmatrix}$$

We can now show that all 6 elements in the upper triangle of the **RHS** equal those in the upper triangle of the **LHS**, thus proving the symmetric matrices are equivalent:

$$\text{RHS Term 1:} \quad \frac{\partial \ell^2}{\partial^2 w_1^2} = \frac{\partial}{\partial w_1} x_1^i \cdot \sigma(s) - y^i \cdot x_1^i = \sigma(s)' x_1^i \cdot \frac{\partial s}{\partial w_1} = \sigma(s)'(x_1^i)^2 \quad \boxed{\text{Matches LHS}}$$

$$\text{RHS Term 2:} \quad \frac{\partial \ell^2}{\partial^2 w_1 w_2} = \frac{\partial}{\partial w_2} x_1^i \cdot \sigma(s) - y^i \cdot x_1^i = \sigma(s)' x_1^i \cdot \frac{\partial s}{\partial w_2} = \sigma(s)' x_1^i x_2^i \quad \boxed{\text{Matches LHS}}$$

$$\text{RHS Term 3:} \quad \frac{\partial \ell^2}{\partial^2 w_1 b} = \frac{\partial}{\partial b} x_1^i \cdot \sigma(s) - y^i \cdot x_1^i = \sigma(s)' x_1^i \cdot \frac{\partial s}{\partial b} = \sigma(s)' x_1^i \quad \boxed{\text{Matches LHS}}$$

$$\text{RHS Term 4:} \quad \frac{\partial \ell^2}{\partial^2 w_2^2} = \frac{\partial}{\partial w_2} x_2^i \cdot \sigma(s) - y^i \cdot x_2^i = \sigma(s)' x_2^i \cdot \frac{\partial s}{\partial w_2} = \sigma(s)'(x_2^i)^2 \quad \boxed{\text{Matches LHS}}$$

$$\text{RHS Term 5:} \quad \frac{\partial \ell^2}{\partial^2 w_2 b} = \frac{\partial}{\partial b} x_2^i \cdot \sigma(s) - y^i \cdot x_2^i = \sigma(s)' x_2^i \cdot \frac{\partial s}{\partial b} = \sigma(s)' x_2^i \quad \boxed{\text{Matches LHS}}$$

$$\text{RHS Term 6:} \quad \frac{\partial \ell^2}{\partial^2 b^2} = \frac{\partial}{\partial b} \sigma(s) - y^i = \sigma(s)' \cdot \frac{\partial s}{\partial b} = \sigma(s)' \cdot 1 = \sigma(s)' \quad \boxed{\text{Matches LHS}}$$

We have successfully shown that the two symmetric matrices are equivalent.

(b) Furthermore, show that $\nabla_\theta^2 \ell^i(\theta)$ is positive semi-definite

Using the properties from question 2.1.a. we now know that the Hessian can be expressed in the following form:

$$\nabla_\theta^2 \ell^i(\theta) = \mathbf{c} x x^T$$

We can now easily show that all eigenvalues are non-negative, as a matrix of the form xx^T is always a rank-1 matrix with eigenvalues equivalent to $\lambda_1 = \|\mathbf{v}\|^2$ and $\lambda_2 = \lambda_3 = 0$. Using the properties of norm, we know that $\|\mathbf{v}\|^2$ is always non-negative, thus all eigenvalues are non-negative and the Hessian is positive semi-definite.

Problem 2.3.

(a) Show that the total loss function is convex

We can show the total loss function is PSD for all θ, which is sufficient evidence that the total loss function is convex:

Linearity of the Hessian: $\nabla_\theta^2 L(\theta) = \frac{1}{N} \sum_{i=1}^N \nabla_\theta^2 l^i(\theta)$

$$\Longrightarrow \text{The Hessian of the total loss function is the sum of } i \text{ PSD Hessians}$$

$$\Longrightarrow \text{The total loss function has a PSD Hessian}$$

A twice-differentiable function is convex if and only if its Hessian is PSD for all θ, which we have shown is true. The total loss function is convex.

(b) Show that the total loss function is strictly convex provided the feature matrix

$$\begin{bmatrix} x_1^1 & x_2^1 & 1 \\ x_1^2 & x_2^2 & 1 \\ & \vdots & \\ x_1^n & x_2^n & 1 \end{bmatrix}$$

Problem 2.3

(b) Show that the total loss function is strictly convex provided the feature matrix X has full rank.

To show that $L(\theta)$ is strictly convex, we must demonstrate that its Hessian $\nabla_\theta^2 L(\theta)$ is **positive definite** (PD) for all θ. This implies that for any non-zero vector $\mathbf{v} \in \mathbf{R}^3$, the quadratic form $\mathbf{v}^T \nabla_\theta^2 L(\theta)\mathbf{v}$ is strictly greater than zero.

From our previous derivations, the total Hessian is the average of the per-sample Hessians:

$$\nabla_\theta^2 L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sigma'(s_\theta(x^i))\mathbf{x}_i \mathbf{x}_i^T$$

This summation can be expressed in matrix form as:

$$\nabla_\theta^2 L(\theta) = \frac{1}{N} X^T D X$$

Where:

- X is the $N \times 3$ feature matrix: $X = \begin{bmatrix} x_1^1 & x_2^1 & 1 \\ & \vdots & \\ x_1^N & x_2^N & 1 \end{bmatrix}$.
- D is an $N \times N$ diagonal matrix where $D_{ii} = \sigma'(s_\theta(x^i)) = \sigma(s^i)(1 - \sigma(s^i))$.

Proof of Positive Definiteness:

1. **D is Positive Definite:** Since the sigmoid function outputs $0 < \sigma(s) < 1$, the derivative $\sigma(s)(1-\sigma(s))$ is strictly positive for all real s. Thus, D is a diagonal matrix with strictly positive entries, making it PD.
2. **Full Rank Condition:** If the feature matrix X has **full column rank** ($rank = 3$), then for any non-zero vector $\mathbf{v}$, the product $X\mathbf{v} \neq 0$.
3. **Quadratic Form:** Letting $\mathbf{z} = X\mathbf{v}$, we have:

$$\mathbf{v}^T (X^T D X)\mathbf{v} = (X\mathbf{v})^T D (X\mathbf{v}) = \mathbf{z}^T D \mathbf{z}$$

   Since D is PD and $\mathbf{z} \neq 0$, it follows that $\mathbf{z}^T D \mathbf{z} > 0$.

Since the Hessian is positive definite, the total loss function $L(\theta)$ is **strictly convex**, which guarantees that the global minimum is unique.

(c) Any local minimizer of the total loss function, if it exists, is a global minimizer.

For any convex function defined on a convex set, any local minimizer is also a global minimizer. We have proved that the total loss function is strictly convex given a feature matrix with a full rank, which is sufficient to state that this condition holds.

Problem 2.4.

Show that the total loss function is always positive. That is for all parameters θ and training data $\{x^i, y^i\}_{i=1}^N$, $L(\theta) > 0$.

The per-sample loss is defined as:

$$\ell_\theta^i(\theta) = \log(1 + e^{s_\theta(x^i)}) - y^i(s_\theta(x^i))$$

Where $s_\theta(x^i) = w_1 x_1^i + w_2 x_2^i + b$, and $y^i \in \{0, 1\}$ for binary outcomes. We can evaluate the two possible cases for y^i:

- **Case 1:** $y^i = 1$
  The loss simplifies to:
  $$\ell^i(\theta) = \log(1 + e^{s_\theta(x^i)}) - s_\theta(x^i)$$
  Since $1 + e^{s_\theta(x^i)} > e^{s_\theta(x^i)}$ and the natural logarithm is increasing, we have:
  $$\log(1 + e^{s_\theta(x^i)}) > \log(e^{s_\theta(x^i)}) = s_\theta(x^i)$$
  Substituting this back into the loss equation:
  $$\ell^i(\theta) > s_\theta(x^i) - s_\theta(x^i) = 0 \implies \ell^i(\theta) > 0$$

- **Case 2:** $y^i = 0$
  The loss simplifies to:
  $$\ell^i(\theta) = \log(1 + e^{s_\theta(x^i)})$$
  Since the exponential function $e^{s_\theta(x^i)}$ is strictly positive for all real inputs, it follows that $1+e^{s_\theta(x^i)} > 1$. Consequently:
  $$\log(1 + e^{s_\theta(x^i)}) > \log(1) = 0 \implies \ell^i(\theta) > 0$$

Since the per-sample loss $\ell^i(\theta)$ is strictly positive for every data point i, the sum of these positive values must also be positive. Therefore, the total loss function, which is the mean of these terms, satisfies:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell^i(\theta) > 0$$

This holds for all finite parameters θ.

Problem 2.5.

Suppose the data $(x^i, y^i)_{i=1}^N$ are linearly separable... Show that $\lim_{t\to+\infty} L(t\tilde{\theta}) = 0$.

Let the scaled parameter be $t\tilde{\theta}$. The score for each sample becomes $s_{t\tilde{\theta}}(x^i) = t\tilde{s}(x^i)$. The total loss function evaluated at this scaled parameter is:

$$L(t\tilde{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \left[\log(1 + e^{t\tilde{s}(x^i)}) - y^i(t\tilde{s}(x^i))\right]$$

To evaluate the limit as $t \to +\infty$, we analyze the term inside the summation for the two possible values of y^i.

- **Case 1:** $y^i = 1$
  By the linear separability assumption, if $y^i = 1$, then $\tilde{s}(x^i) > 0$. As $t \to +\infty$, the term $t\tilde{s}(x^i) \to +\infty$. We can rewrite the loss term for this case:
  $$\ell^i = \log(1 + e^{t\tilde{s}(x^i)}) - t\tilde{s}(x^i)$$

Factoring out $e^{t\tilde{s}(x^i)}$ inside the logarithm:

$$\ell^i = \log(e^{t\tilde{s}(x^i)}(e^{-t\tilde{s}(x^i)} + 1)) - t\tilde{s}(x^i)$$

$$\ell^i = \log(e^{t\tilde{s}(x^i)}) + \log(1 + e^{-t\tilde{s}(x^i)}) - t\tilde{s}(x^i)$$

$$\ell^i = t\tilde{s}(x^i) + \log(1 + e^{-t\tilde{s}(x^i)}) - t\tilde{s}(x^i) = \log(1 + e^{-t\tilde{s}(x^i)})$$

Taking the limit:

$$\lim_{t\to+\infty} \log(1 + e^{-t\tilde{s}(x^i)}) = \log(1 + 0) = 0$$

- **Case 2:** $y^i = 0$
  By the assumption, if $y^i = 0$, then $\tilde{s}(x^i) < 0$. As $t \to +\infty$, the term $t\tilde{s}(x^i) \to -\infty$. The loss term for this case simplifies to:

  $$\ell^i = \log(1 + e^{t\tilde{s}(x^i)}) - 0 \cdot (t\tilde{s}(x^i)) = \log(1 + e^{t\tilde{s}(x^i)})$$

  Taking the limit:

  $$\lim_{t\to+\infty} \log(1 + e^{t\tilde{s}(x^i)}) = \log(1 + 0) = 0$$

Since the limit of the loss for every individual sample is 0, the limit of the average sum is also 0:

$$\lim_{t\to+\infty} L(t\tilde{\theta}) = \frac{1}{N}\sum_{i=1}^{N} 0 = 0$$

Problem 2.6.

When the data are linearly separable, show that the total loss function does not admit a minimizer.

By contradiction, using the results from Problems 2.4 and 2.5:

1. From Problem 2.4, we established that for any finite parameter set θ, the loss function is strictly positive:

   $$L(\theta) > 0, \quad \forall\theta \in R^d$$

2. From Problem 2.5, we established that if the data is linearly separable, there exists a direction $\tilde{\theta}$ such that:

   $$\lim_{t\to+\infty} L(t\tilde{\theta}) = 0$$

   This implies that the infimum of the loss function is 0:

   $$\inf_{\theta} L(\theta) = 0$$

3. **Non-Existence of Minimizer:** Suppose there exists a minimizer θ^* that achieves the minimum loss value. By the strict positivity condition (1), this minimum value must be strictly greater than 0:

   $$L(\theta^*) = \epsilon > 0$$

   However, by property (2), we can find a scaled parameter $t\tilde{\theta}$ with sufficiently large t such that:

   $$L(t\tilde{\theta}) < \epsilon$$

   This contradicts the assumption that θ^* is a minimizer (since we found a value lower than $L(\theta^*)$).

Therefore, the function can become arbitrarily close to 0 by scaling the weights towards infinity in the direction of the separating hyperplane, but it never attains the value 0 at any finite θ. Thus, no minimizer exists.

AM 230 - Course Project I (Week 2)
Binary Outcome Prediction

Luke Catalano
UC Santa Cruz, M.S. Quantitative Economics

January 2026

Problem 3.1.

In this problem, you will implement gradient descent for the logistic regression loss (6) using a fixed step size. A template implementation of gradient descent is provided in

solvers/solve_gd.m

The objective function (loss and gradient) is implemented in

models/logistic_objective.m

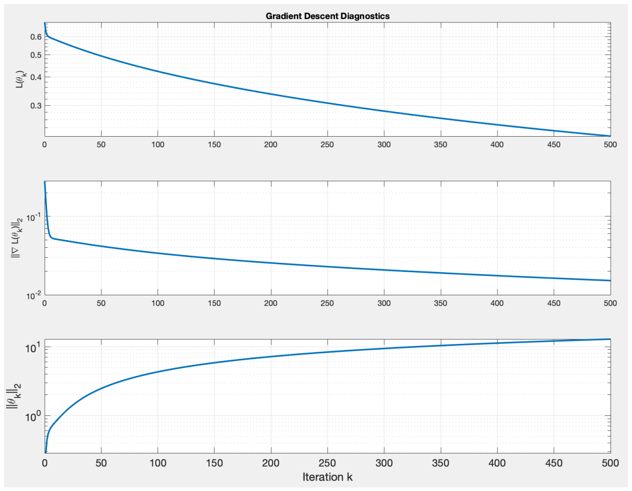
You are not expected to implement the loss or its derivatives from scratch. Complete the TODO parts in **solve_gd.m** so that the solver correctly implements the gradient descent iteration

$$\theta_{k+1} = \theta_k - \alpha \nabla L(\theta_k)$$

where $\alpha > 0$ is a fixed step size specified by **opts.alpha_fixed**. Your implementation should include:

- Computation of the descent direction
- The parameter update with fixed step size
- A stopping criterion based on the gradient norm

Implementation output:



Code updates:

```
p     = -g; % Negative gradient descent direction implemented
theta = theta + alpha*p; % theta is updated with the descent direction times the step size
```

Problem 3.2.

Use your **solve_gd.m** from the previous problem to apply gradient descent with initial condition

$$\theta = [w_1, w_2, b]^T = [0, 0, 0]^T$$

and constant step size $\alpha = 1$

(a) Iterate gradient descent for k= 500 iterations. Report your final parameter values and check if the result achieves perfect classification. If not, report how many samples are misclassified. Note that perfect classification refers to zero training misclassification, not necessarily zero loss.

After 500 iterations:

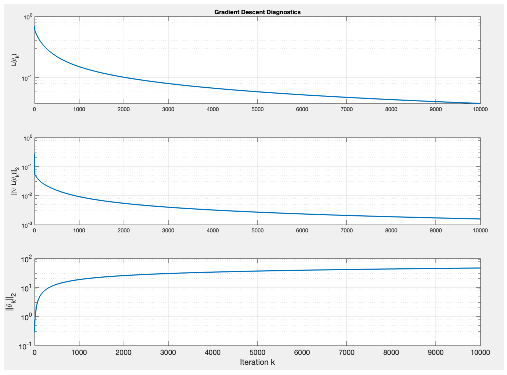
$$[w_1, w_2, b] = [4.1799, 9.1322, -8.0352]$$

The model classified 96.69% of samples, missing only 1.

(b) With more iterations, can you obtain a result that classifies all samples correctly?

For linearly separable data, gradient descent can drive the loss arbitrarily close to zero but cannot converge to a finite minimizer. Therefore, while more iterations will lower the loss, the parameter vector will continue to grow. This does not mean that we cannot achieve perfect classification, however. It is certainly possible to obtain perfect classification with an increase in iterations.

(c) Set the number of iterations to k = 10000. Plot the loss L(θ), the 2-norm of the gradient $||\nabla_\theta L||$, and the 2-norm of the parameter $||\theta||$ versus iteration k. Choose log scale for better visualizations. Report what you observe. Does the loss and the gradient appear to approach 0? Does the parameter norm $||\theta||$ stay bounded or grow large?



Loss $L(\theta)$: The loss continues to decrease toward zero but at a very slow rate. It does not "bottom out" at a finite value.

Gradient Norm $\|\nabla L\|_2$: The norm is approaching 10^{-3} but has not yet reached the tolerance of 10^{-8}. This indicates that the solver is still "descending," but the "bottom" is at infinity.

Parameter Norm $\|\theta\|_2$: The norm is steadily growing and has reached nearly 10^2. It shows no sign of staying bounded.

(d) Relate your observations to the theoretical study in the previous Homework problems.

The observations in part (c), specifically the vanishing loss and the unbounded growth of the parameter norm $\|\theta\|_2$, provide empirical evidence for the concepts form Problem 2.6. Because the training data is linearly separable, the loss function (6) does not admit a finite minimizer θ^*

AM 230 - Course Project I (Week 3)
Binary Outcome Prediction

Luke Catalano
UC Santa Cruz, M.S. Quantitative Economics

February 2026

A common approach to ensure the existence of a minimizer of the logistic regression loss (6) is to add a quadratic regularization term to prevent the parameters from increasing to infinity. The modified loss function to minimize is

$$\bar{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(log\left(1+e^{s_\theta(x^i)}\right) - y^i s_\theta(x^i)\right) + \mu||\theta||^2$$

where $\mu > 0$ is a fixed regularization constant.

Problem 4.1

Compute the gradient and the Hessian of the loss function (9) and compare them to the gradient and the Hessian of the original loss function (3).

We begin by computing the gradient:

$$\nabla\bar{L}(\theta) = \frac{\partial\bar{L}(\theta)}{\partial\theta} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{e^{s_\theta(x^i)}}{1+e^{s_\theta(x^i)}}x^i - y^i x^i\right) + 2\mu\theta$$

$$\nabla\bar{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\underbrace{(\sigma(\theta^T x^i) - y^i)}_{\text{Scalar Error}}\underbrace{x^i}_{\text{Vector direction}} + 2\mu\theta$$

Differentiating the Sigmoid: To get the Hessian, we take the derivative of our gradient. The key part is the derivative of $\sigma(\theta^T x^i)$:

The derivative of $\sigma(z)$ is $\sigma(z)(1-\sigma(z))$.

Applying the chain rule again, we multiply by $(x^i)^T$ (the transpose of the feature vector).

$$\nabla^2\bar{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\sigma(\theta^T x^i)(1-\sigma(\theta^T x^i))x^i(x^i)^T\right] + \frac{\partial}{\partial\theta}(2\mu\theta)$$

Differentiating the Penalty: The derivative of the vector $2\mu\theta$ with respect to θ is the Identity Matrix I scaled by 2μ.

$$\nabla^2\bar{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\sigma(\theta^T x^i)(1-\sigma(\theta^T x^i))x^i(x^i)^T\right] + 2\mu I$$

Problem 4.2

Show that the loss function (9) is 2μ-strongly convex. Thus, it has a unique global minimizer.

A twice-differentiable function f is m-strongly convex iff $\nabla^2 f(\theta) \succeq mI$ for all θ, meaning the Hessian is positive definite with all $eig(\nabla^2 f(\theta)) \geq m$. Thus we must show: $\nabla^2 \bar{L}(\theta) \succeq 2\mu I \ \forall\theta$.

From 4.1 we have

$$\nabla^2 \bar{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \sigma(\theta^T x_i)\left(1-\sigma(\theta^T x_i)\right) x_i x_i^T \ + \ 2\mu I.$$

Now we show the first term is positive semidefinite (PSD):

For any vector v:

$$v^T(x_i x_i^T)v = (x_i^T v)^2 \geq 0 \implies x_i x_i^T \succeq 0$$

It also holds that:

$$\sigma(z)(1-\sigma(z)) \geq 0 \text{ for all z, because } \sigma(z) \in (0,1).$$

Therefore each term

$$\sigma(\theta^T x_i)(1-\sigma(\theta^T x_i))\, x_i x_i^T \succeq 0$$

It holds that sums/averages of PSD matrices stay PSD, so:

$$\frac{1}{N}\sum_{i=1}^{N} \sigma(\theta^T x_i)(1-\sigma(\theta^T x_i))\, x_i x_i^T \succeq 0.$$

Thus:

$$\nabla^2 \bar{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \sigma(\theta^T x_i)(1-\sigma(\theta^T x_i))\, x_i x_i^T + 2\mu I \succeq 0 + 2\mu I = 2\mu I.$$

Which is the condition for 2μ-strong convexity. A strongly convex function has at most one minimizer; equivalently, strong convexity implies the objective is strictly convex. So $\bar{L}(\theta)$ has a unique global minimizer.

Problem 4.3

Implement gradient descent with constant step size α for minimizing the modified loss function (9). Set the initial values

$$\theta = [w_1, w_2, b]^T = [0, 0, 0]^T$$

and constant step size $\alpha = 1$. Compare the performance on three different penalty parameters:
$\mu = 10^{-2}, \mu = 10^{-3}$, and $\mu = 10^{-4}$. This can be achieved by assigning ***problem.mu*** in the main script. Do you achieve perfect classification? Does the norm of the parameter grow to infinity as in Problem 3.2 (with no penalty)?

- problem.mu $= 10^{-2}$: We achieve **66.67% training accuracy,** with 1**0 misclassified samples.** The norm of the parameter does not grow to infinity..
- problem.mu $= 10^{-3}$: We achieve **90.00% training accuracy**, with **3 misclassified samples**. The norm of the parameter does not grow to infinity..
- problem.mu $= 10^{-4}$: We achieve **96.67% training accuracy,** with **1 misclassified samples.** The norm of the parameter does not grow to infinity.

We do not achieve perfect classification, and the parameter norm increases initially from zero and then levels off, converging to a finite value due to the ℓ_2 regularization. This contrasts with the unregularized case in Problem 3.2, where the norm diverges."

AM 230 - Course Project I (Week 5)

Binary Outcome Prediction

Luke Catalano

M.S. Quantitative Economics

February 2026

Problem 5.1

(Constant step size) In this problem, we choose $\mu = 10 - 4$ and apply gradient descent with a constant step size $\alpha = 10$, i.e. setting: problem.mu = 1e-4 and opts.alpha fixed = 10.

(a) Plot the norm of the gradient $||\nabla \bar{L}(\theta_k)||$ on a log scale versus the number of iterations. Do you observe a linear rate of convergence? If yes, provide an estimation of the constant r.

$$||\nabla \bar{L}(\theta_k)|| \leq r||\nabla \bar{L}(\theta_{k-1})||$$

Yes. After the initial "zig-zag" phase (roughly k ¿ 80), the plot of $\|\nabla \bar{L}(\theta_k)\|$ versus iterations on a log scale is approximately linear, indicating linear convergence. Estimating the ratio using two points in the linear region gives $r \approx 0.99$.

(b) What are the eigenvalues of the Hessian matrix at the numerical minimizer θ^* that you computed? Does the linear convergence rate r you computed relate to the eigenvalues of the Hessian as we theoretically analyzed in the lecture for quadratic functions?

At the numerical minimizer θ^*, the Hessian eigenvalues are approximately $(4\times10^{-4},\ 3.6\times10^{-3},\ 1.249\times10^{-1})$. For gradient descent with constant step size α, the local linear convergence factor is $r = \max_i |1-\alpha\lambda_i|$. Using $\alpha = 10$, this predicts $r_{\text{theory}} = 0.9956$, which is consistent with the approximately linear (straight-line on a semilog plot) decay of $\|\nabla \bar{L}(\theta_k)\|$ observed in part (a). Since $\alpha\lambda_{\max} = 1.249 < 2$, the method is stable, and the moderate condition number $\kappa \approx 2.8 \times 10^2$ helps explain the relatively slow rate.

(c) When α is sufficiently large, the convergence cannot be guaranteed. Can you estimate this threshold and numerically verify it?

Yes. For (locally) strongly convex $\bar{L}$ near θ^*, constant–step GD converges only if $0 < \alpha < \frac{2}{\lambda_{\max}(H(\theta^*))}$.

$$\lambda_{\max} \approx 0.1249 \implies \alpha_{\text{crit}} \approx \frac{2}{0.1249} \approx 16.01.$$

Step-size stability test (predicted $\alpha_{\text{crit}} \approx 16.01$)

α	f_{end}	$\|\nabla L\|_{\text{end}}$	Status
15.00	1.667×10^{-1}	9.955×10^{-9}	Converged
15.50	1.667×10^{-1}	9.967×10^{-9}	Converged
16.00	1.667×10^{-1}	9.998×10^{-9}	Converged
16.50	1.675×10^{-1}	1.393×10^{-2}	Not converged
17.00	1.711×10^{-1}	3.082×10^{-2}	Not converged

(d) With a constant step size, what is the best convergence rate? What is the corresponding step size? Numerically verify your result.

For constant–step GD on a (locally) strongly convex quadratic model with Hessian eigenvalues in $[\lambda_{\min}, \lambda_{\max}]$, the best (fastest) linear rate you can get by tuning α is obtained by minimizing $r(\alpha) = \max_i |1 - \alpha\lambda_i|$. The optimal constant step size is:

$$\alpha^* = \frac{2}{\lambda_{\min} + \lambda_{\max}}$$

and the resulting best linear convergence factor is:

$$r^* = \frac{\kappa - 1}{\kappa + 1} \quad \text{where} \quad \kappa = \frac{\lambda_{\max}}{\lambda_{\min}} \quad \text{and} \quad \lambda_{\min} = 0.0004, \quad \lambda_{\max} = 0.1249$$

Thus α^* can be derived as follows:

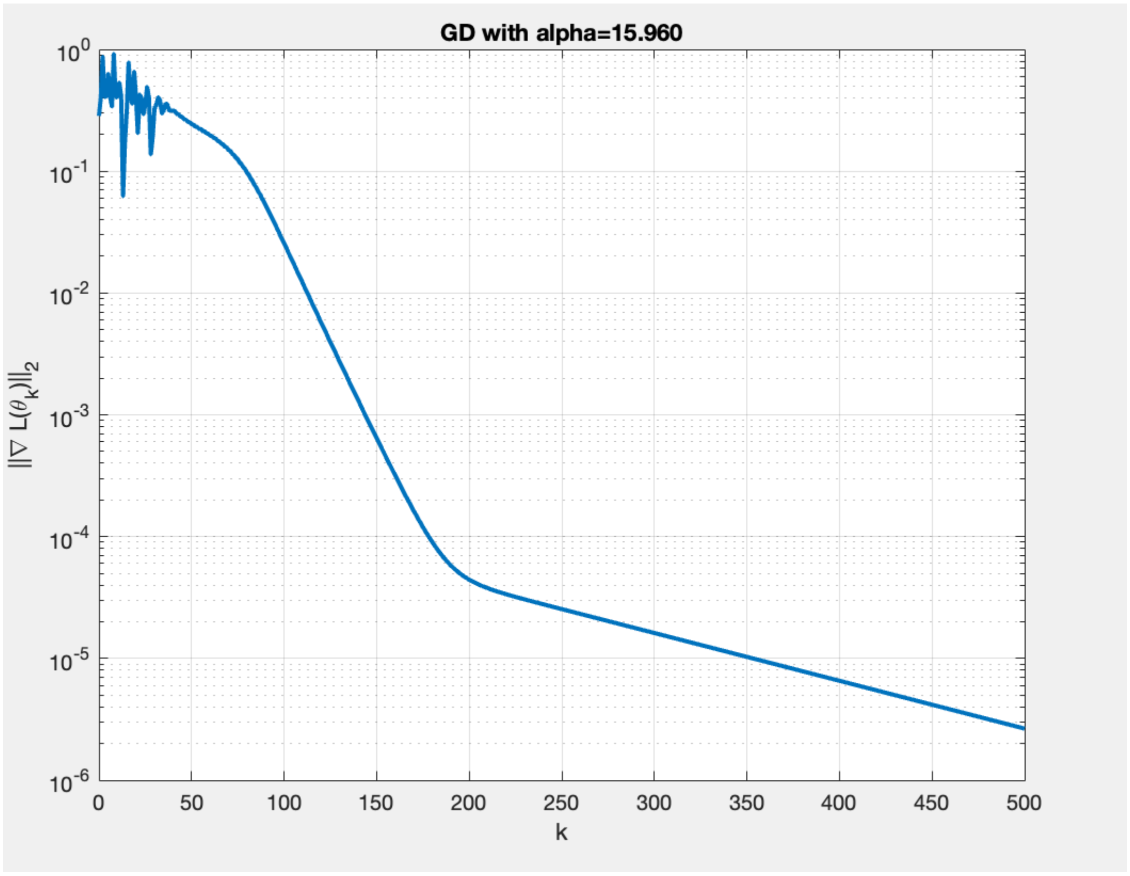
$$\alpha^* = \frac{2}{0.0004 + 0.1249} = \frac{2}{0.1253} \approx 15.96.$$

And r^* can be computed as well:

$$\text{First,} \quad \kappa = \frac{0.1249}{0.0004} = 312.25 \quad \text{Then} \quad r^* = \frac{312.25 - 1}{312.25 + 1} = \frac{311.25}{313.25} \approx 0.9936.$$

$$\boxed{\alpha^* \approx 15.96, \qquad r^* \approx 0.9936}$$

The empirically fastest convergence occurs near $\alpha \approx 16$, matching the theoretical optimal step size α^*, and the observed linear decay of the gradient norm is consistent with the predicted rate $r^* \approx 0.994$.



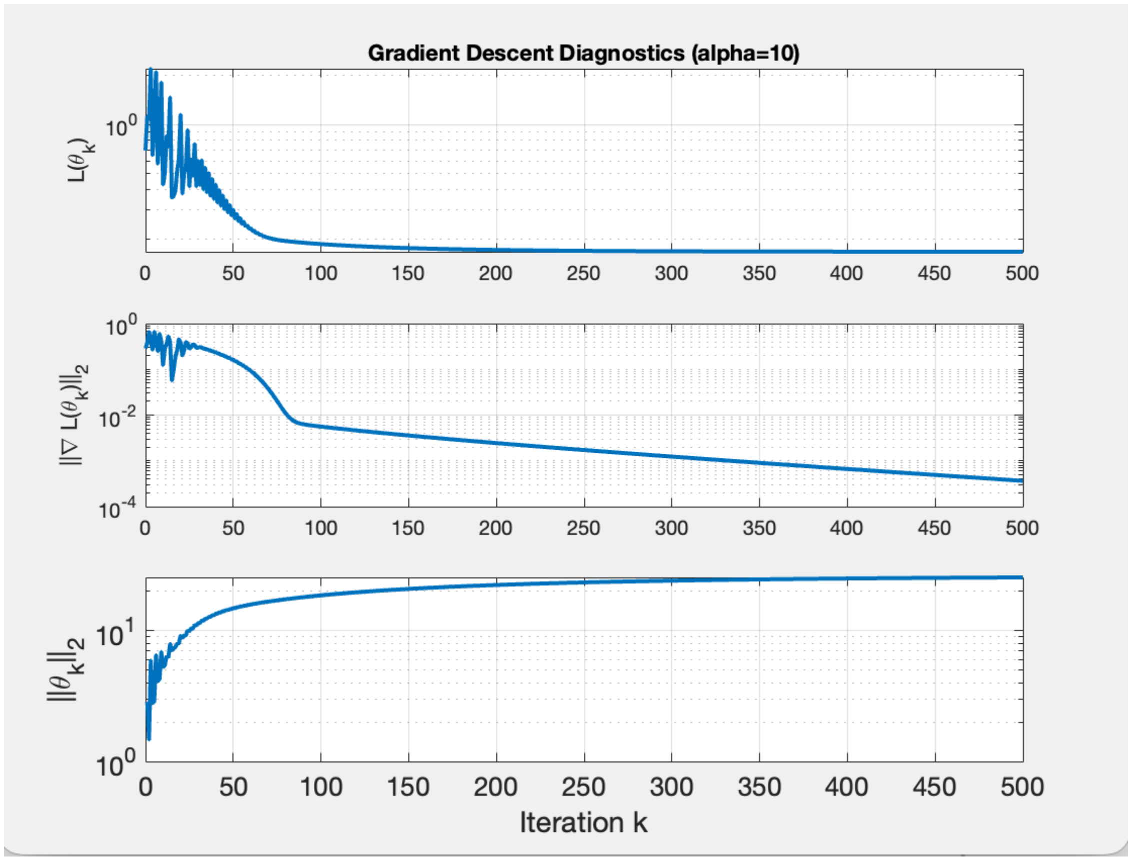
Problem 5.2

(Variable step sizes) In this experiment, we study gradient descent with variable step sizes and compare its performance with gradient descent using the optimal constant step size $\alpha = \alpha^*$ derived in Problem 5.1.

(a) Modify the gradient descent solver solve gd.m so that:

- when opts.useLineSearch = false, a fixed step size α = opts.alpha fixed is used;
- when opts.useLineSearch = true, the step size is computed by calling steplength with input opts.ls. A set of default line search parameters is provided in the main script:

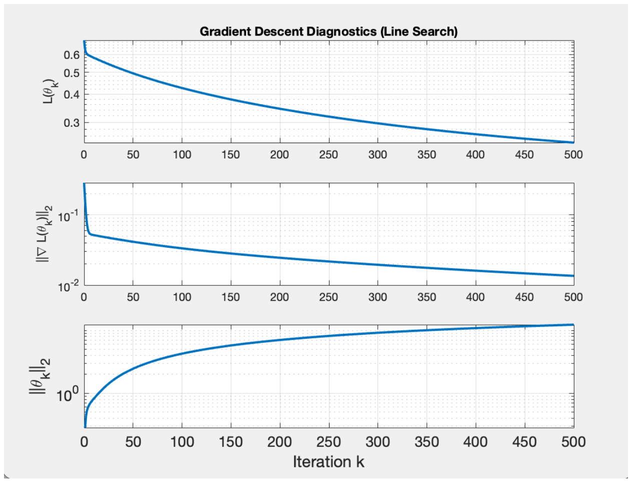
```
opts.ls = struct( ...
    'c1', 1e-4, ...
    'c2', 0.9, ...
    'alpha0', 1.0, ...
    'alpha_max', 100, ...
    'alpha_min', 1e-6, ...
    'maxIter', 50, ...
    'maxZoom', 50 );
```



The generated figure shows the convergence diagnostics for gradient descent with fixed step size $\alpha = 10$. The objective value and the gradient norm decrease monotonically, indicating convergence of the algorithm. The gradient norm exhibits an approximately linear decay on the log scale after the initial transient iterations, suggesting a linear convergence. The parameter norm stabilizes as the iterates approach the numerical minimizer.

(b) Run gradient descent with $\mu = 10^{-4}$, $\theta_0 = 0$ (as in the previous problem), and using the default line search parameters. Report the convergence behavior of $||\nabla \bar{L}(\theta_k)||$

.

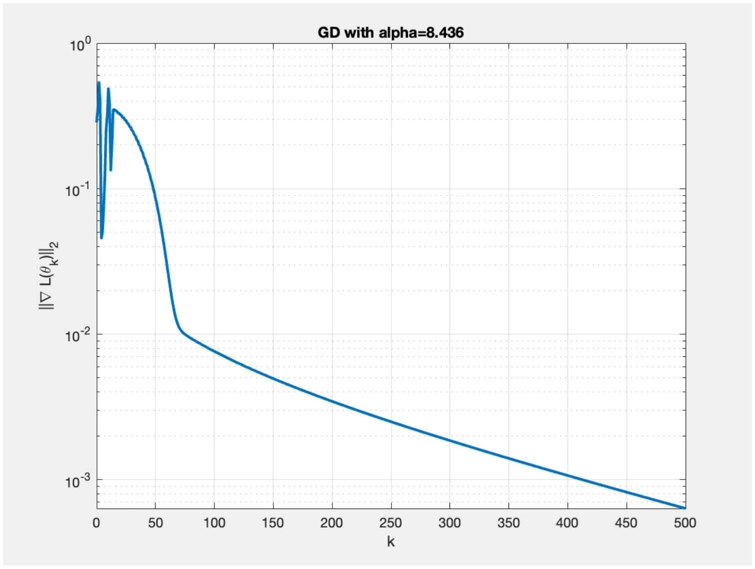


Using the default line search parameters, the gradient norm decreases monotonically but at a noticeably slower rate compared to the optimal fixed step size. The convergence appears sublinear initially due to conservative step-size selection by the backtracking procedure, though eventual linear decay becomes visible in later iterations.

(c) Implement gradient descent with the approximate "exact" line search described above with $\mu = 10^{-4}$ and the same initial condition $\theta_0 = 0$. Set the line search parameters

$$c_1 = 10^{-8}, \qquad c_2 = 10^-7$$

Plot the norm of the gradient $||\nabla \bar{L}(\theta_k)||$ versus iteration k (in log scale) and report your observations.



After the initial oscillations (first 50 iterations), the curve becomes approximately linear on the log scale. This indicates linear convergence of gradient descent with the approximate "exact" line search. Compared to fixed step size runs, the method automatically selects step sizes (here around 8.4 initially) that produce stable and efficient decrease of the gradient norm. No divergence or instability appears, confirming the effectiveness of the line search.

(d) Compare the convergence behavior of gradient descent with optimal constant step size α^*,, gradient descent with strong Wolfe line search, gradient descent with approximate "exact" line search. Discuss the results in terms of convergence speed and computational cost.

We compare three variants of gradient descent:

- **Optimal constant step size α^*:** With $\alpha^* = \frac{2}{\lambda_{\min}+\lambda_{\max}}$, gradient descent has the fastest linear convergence per iteration among the three methods. On the log-scale plot of $\|\nabla \bar{L}(\theta_k)\|_2$, the slope is steepest after the non-smooth phase. Computationally, this method is cheapest per iteration since each step requires only one gradient evaluation and no additional function evaluations for step-size selection.

- **Strong Wolfe line search:** Gradient descent with strong Wolfe line search is robust and guarantees a principled step size satisfying sufficient decrease and curvature conditions. In practice, it often converges reliably but can be slower per iteration than the optimal fixed-step method because the line search may accept conservative step sizes. Moreover, its computational cost per iteration is higher because each iteration may require multiple objective/gradient evaluations during the line search procedure.

- **Approximate "exact" line search ($c_1 = 10^{-8}$, $c_2 = 10^{-7}$):** The approximate "exact" line search produces stable convergence and, after a short non-smooth phase, the log-scale gradient norm decays approximately linearly, indicating linear convergence. Empirically, the chosen step sizes tend to be close to a near-optimal constant step (e.g., $\alpha \approx 8.4$ in our run), yielding convergence behavior similar to the optimal fixed-step method. However, it still incurs additional computational overhead from performing line search evaluations each iteration.

The optimal fixed step size α^* provides the best convergence speed per unit computation when curvature information (via Hessian eigenvalues) is available. Line search methods (strong Wolfe and approximate "exact") trade additional per-iteration cost for robustness and automatic step-size selection; the approximate "exact" line search can approach fixed-step performance when it selects step sizes near α^*.

Problem 5.3

(Conditioning versus regularization strength μ) In this problem we investigate how the quadratic regularization parameter μ affects the conditioning of the regularized logistic regression problem and the convergence rate of gradient descent.

(a) Consider several values of the regularization parameter:

$$\mu = \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$$

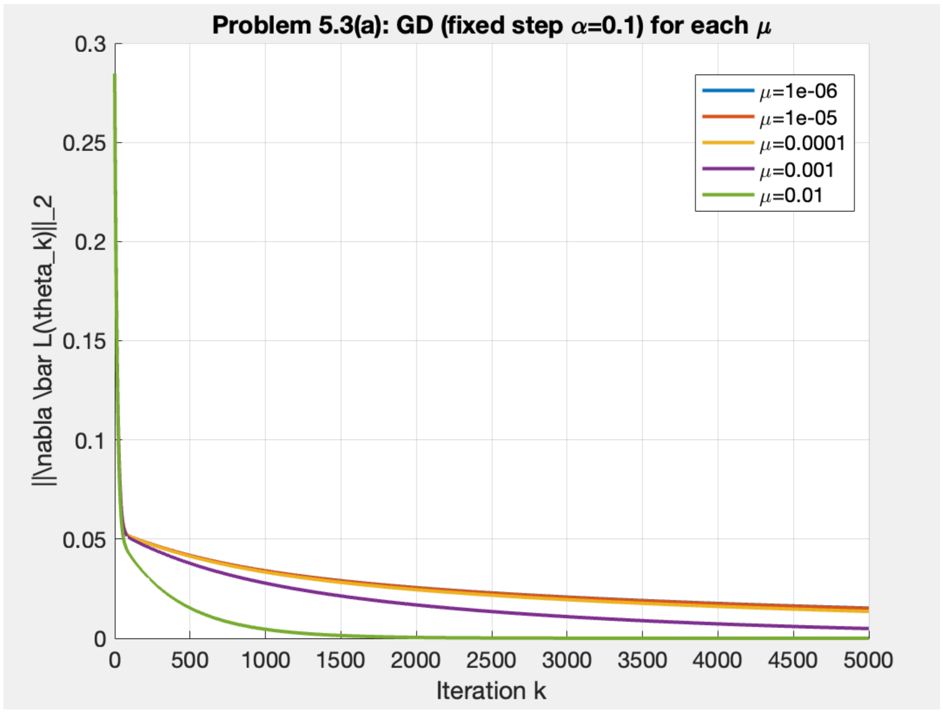
For each value of μ

- Implement gradient descent with inexact line search using the strong Wolfe conditions with default options, and plot the norm of the gradient $||\nabla \bar{L}(\theta_k)||$ in log scale versus the number of iterations.
- Report the numerical minimizer $\theta^*(\mu)$.
- Evaluate the Hessian matrix at the optimal solution

$$H(\mu) = \nabla^2 \bar{L}(\theta^*(\mu))$$

and its smallest and largest eigenvalues λ_{min} and λ_{max}. Report the condition number:

$$\kappa(\mu) = \frac{\lambda_{max}}{\lambda_{min}}$$



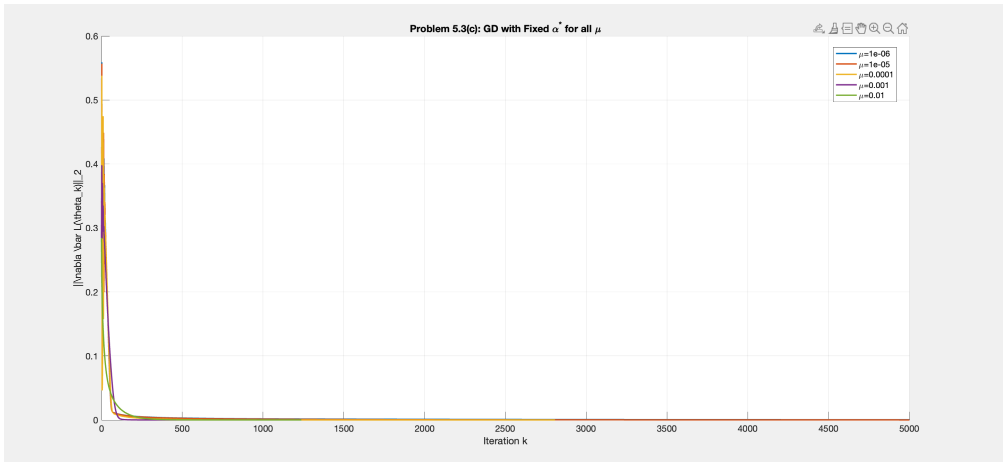
μ	λ_{min}	λ_{max}	$\kappa(\mu)$	Iterations
10^{-6}	0.0013366	0.22913	171.42	5000
10^{-5}	0.0013522	0.22974	169.90	5000
10^{-4}	0.0015073	0.23576	156.41	5000
10^{-3}	0.0029069	0.28403	97.71	5000
10^{-2}	0.0125460	0.38795	30.923	5000

As the regularization parameter μ increases, the smallest eigenvalue of the Hessian increases substantially, while the largest eigenvalue changes more moderately. The condition number decreases significantly, indicating improved conditioning of the optimization problem.

(b) Compare the observed convergence rates for different values of μ. How does improved conditioning influence the convergence speed of gradient descent? What trade-off does regularization introduce between optimization efficiency and model bias?

As the regularization parameter μ increases, the smallest eigenvalue of the Hessian increases, which improves the condition number of the problem. Because gradient descent with a fixed step size is sensitive to conditioning, larger μ values lead to faster convergence, while small μ values result in slower progress. This behavior is clearly visible in the convergence plots, where the curves corresponding to larger μ decrease more rapidly.

(c) For each μ run gradient descent with the optimal constant step size:



When the theoretically optimal constant step size $\alpha^*(\mu)$ is used, the convergence curves for different values of μ become much more similar. The optimal step size automatically accounts for the local curvature of the objective, compensating for differences in conditioning. As a result, the dependence of convergence speed on μ is significantly reduced.

AM 230 - Course Project I (Week 6)
Binary Outcome Prediction

Luke Catalano
M.S. Quantitative Economics

February 2026

6.1 Implementation

A Newton solver (`solve_newton.m`) was implemented supporting both pure Newton updates and damped Newton updates using a strong Wolfe line search.

6.2 Pure Newton

Figure 1 shows the behavior of pure Newton from $\theta_0 = [1,1,1]^T$, $\theta_0 = [2,2,2]^T$, and several random initializations. For some starting points Newton converges very rapidly, while for others the method behaves poorly, showing that pure Newton works best with fast local convergence but does not guarantee global convergence.

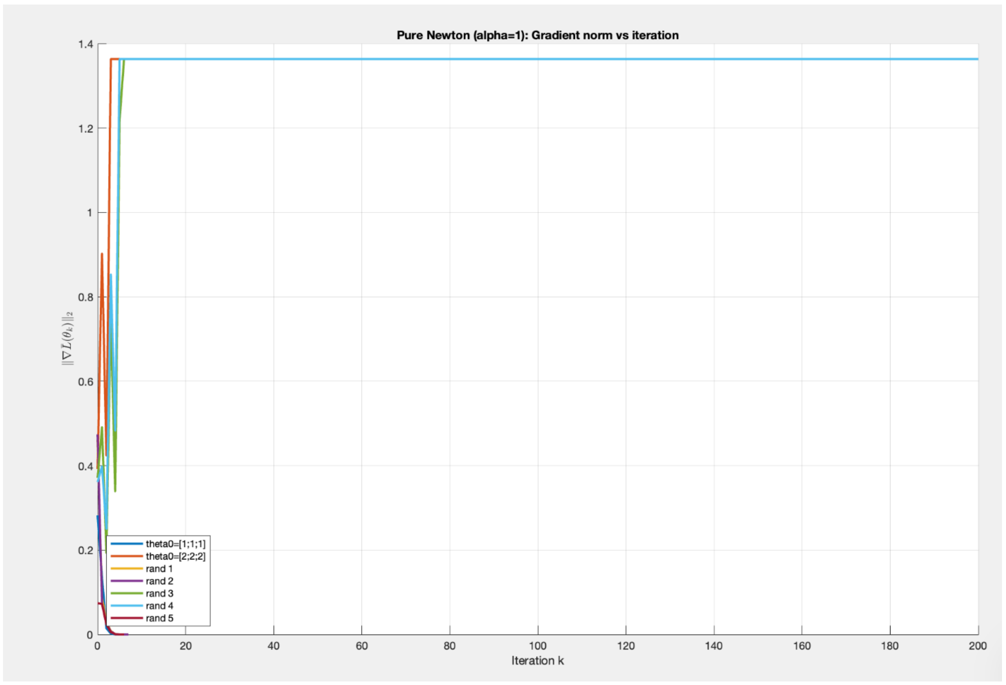


Figure 1: Pure Newton ($\alpha = 1$): gradient norm vs iteration.

6.2(b) Newton vs Gradient Descent

Figure 2 compares damped Newton with gradient descent using the optimal constant step size $\alpha_\star$. Newton's method reaches the desired tolerance in far fewer iterations, illustrating the quadratic local convergence of Newton compared to the linear convergence of gradient descent.

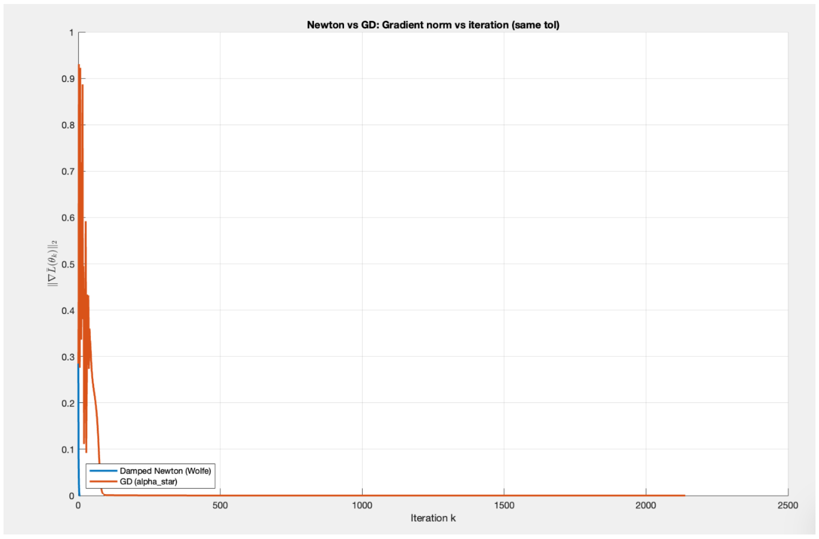


Figure 2: Newton vs Gradient Descent (same tolerance).

6.3 Damped Newton with Strong Wolfe Line Search

6.3(a) Newton Direction is a Descent Direction

The Newton search direction is defined as

$$p_k = -\left(\nabla^2 \bar{L}(\theta_k)\right)^{-1} \nabla \bar{L}(\theta_k).$$

Let $g_k = \nabla \bar{L}(\theta_k)$ and $H_k = \nabla^2 \bar{L}(\theta_k)$. Since the regularized objective is strongly convex, the Hessian H_k is positive definite. Therefore,

$$g_k^T p_k = -g_k^T H_k^{-1} g_k < 0 \quad \text{whenever } g_k \neq 0,$$

because H_k^{-1} is also positive definite. Hence the Newton direction is always a descent direction for $\bar{L}(\theta)$. Consequently, a step size satisfying the strong Wolfe conditions guarantees global convergence of the damped Newton method.

6.3(b,c) Newton's method with adaptive step size

Using a strong Wolfe line search greatly improves global convergence. Figure 3 shows the case where the initial trial step is $\alpha_0 = 1$, while Figure 4 shows the case $\alpha_0 = 0.1$. Both configurations converge reliably from all tested initializations, although $\alpha_0 = 1$ typically requires fewer iterations.

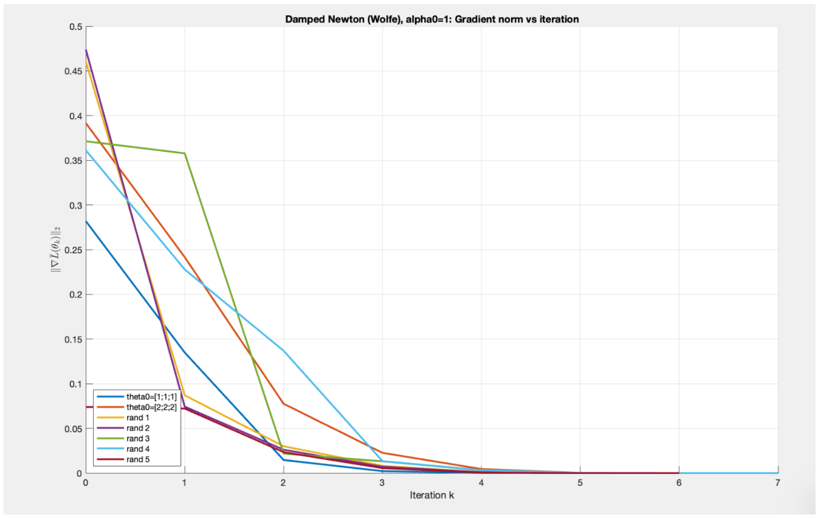


Figure 3: Damped Newton (Wolfe), $\alpha_0 = 1$.

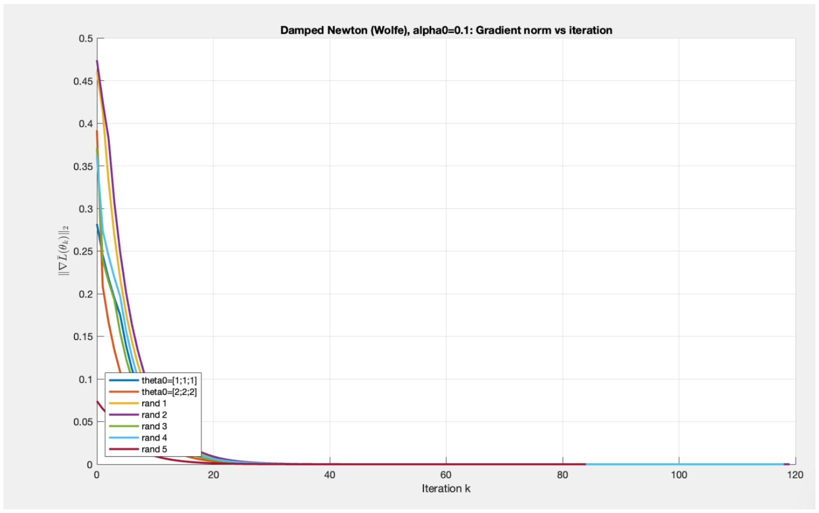


Figure 4: Damped Newton (Wolfe), $\alpha_0 = 0.1$.

6.4 Effect of Conditioning on Newton's Method

Newton's method with strong Wolfe line search was run for several values of the regularization parameter

$$\mu \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}.$$

The results show that Newton's method is significantly less sensitive to conditioning than gradient descent. While changes in μ strongly affect the convergence speed of gradient descent, the number of Newton iterations remains relatively stable across different conditioning levels. This reflects the curvature-correcting nature of Newton's method, which rescales the gradient using second-order information.