COMP 7070 Advanced Topics in AI and ML February 5, 2025

Lecture 3: Regularization and Logistic Regression

Instructor: Yifan Chen Scribes: Haoran Zheng, Hongtao Wang Proof reader: Xiong Peng

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

3.1 Regularization

Regularization is a technique applied in statistical models to encode preference for simpler solutions and prevent overfitting by adding a penalty term to the loss function. This approach helps balance model complexity with predictive performance.

3.1.1 Ridge Regression

Building upon standard linear regression, ridge regression introduces L_2 regularization to enhance model stability. Specifically, it adds a penalty term proportional to the squared magnitude of the coefficients, controlled by a tuning parameter λ:

$$L = \frac{1}{2n}(Y - Xw)^\top (Y - Xw) + \frac{\lambda}{2}\|w\|^2. \tag{3.1}$$

Setting the gradient with respect to w to zero yields the critical equation:

$$\frac{\partial L}{\partial w} = -\frac{1}{n}X^\top (Y - Xw) + \lambda w = 0. \tag{3.2}$$

Solving this system leads to the ridge regression estimator:

$$\hat{w}_\lambda = (X^\top X + n\lambda I)^{-1} X^\top Y. \tag{3.3}$$

3.1.2 Numeric Stability

Having established the mathematical formulation, we now examine its numerical properties. The condition number of a matrix, which determines numerical stability during inversion, plays a crucial role in regression analysis. For ordinary least squares (OLS) with $\lambda = 0$:

$$\kappa(X^\top X) = \frac{\sigma_{\max}^2}{\sigma_{\min}^2}, \tag{3.4}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ denote the largest and smallest singular values of X.

Ridge regression enhances numerical stability through its regularization mechanism. The modified condition number becomes:

$$\kappa\left(X^\top X + n\lambda I\right) = \frac{\sigma_{\max}^2 + n\lambda}{\sigma_{\min}^2 + n\lambda}. \tag{3.5}$$

This modification reduces the condition number by effectively bounding the singular values, making matrix inversion more stable.

3.1.3 MSE Reduction

Beyond numerical stability, ridge regression offers statistical advantages through bias-variance tradeoff. Let us analyze the Mean Squared Error (MSE), defined for an estimator $\hat{w}_\lambda$ as:

$$\text{MSE}(\hat{w}_\lambda) = E[(\hat{w}_\lambda - w^*)^\top(\hat{w}_\lambda - w^*)]. \tag{3.6}$$

Our objective is to demonstrate:

$$\text{MSE}(\hat{w}_\lambda) \leq \text{MSE}(\hat{w}_0). \tag{3.7}$$

Using the trace operator to decompose the expectation:

$$\begin{aligned}\text{MSE}(\hat{w}_\lambda) &= \text{Tr}\left(E[(\hat{w}_\lambda - w^*)(\hat{w}_\lambda - w^*)^\top]\right) \\ &= \text{Tr}(M(\hat{w}_\lambda)).\end{aligned} \tag{3.8}$$

This decomposition leads us to examine the covariance matrix ordering. If $M(\hat{w}_\lambda) \preceq M(\hat{w}_0)$ in Löwner order, then:

$$\text{Tr}(M(\hat{w}_\lambda)) \leq \text{Tr}(M(\hat{w}_0)). \tag{3.9}$$

The proof leverages matrix positivity:

$$\begin{aligned}M(\hat{w}_\lambda) \preceq M(\hat{w}_0) &\iff M(\hat{w}_0) - M(\hat{w}_\lambda) \succeq 0 \\ &\implies \text{Tr}(M(\hat{w}_0)) \geq \text{Tr}(M(\hat{w}_\lambda)).\end{aligned} \tag{3.10}$$

Finally, decomposing the MSE into bias and variance components reveals:

$$M(\hat{w}_\lambda) = \underbrace{\text{Var}(\hat{w}_\lambda)}_{\text{Variance}} + \underbrace{E[(\hat{w}_\lambda - w^*)(\hat{w}_\lambda - w^*)^\top]}_{\text{Bias}^2}. \tag{3.11}$$

This decomposition illustrates how ridge regression achieves lower MSE by optimally balancing increased bias with reduced variance.

3.1.3.1 Bias Vector of Ridge Estimator

Having established the MSE decomposition framework, we now quantitatively analyze the bias component. The bias of an estimator is defined as the difference between its expected value and the true parameter value. For the ridge estimator under the true model $Y = Xw^* + \epsilon$, we derive:

$$\begin{aligned}E(\hat{w}_\lambda - w^*) &= E\left[(X^\top X + n\lambda I)^{-1}X^\top Y\right] - w^* \\ &= (X^\top X + n\lambda I)^{-1}X^\top X w^* - w^* \\ &= \left[(X^\top X + n\lambda I)^{-1}(X^\top X + n\lambda I - n\lambda I)\right] w^* - w^* \\ &= w^* - n\lambda(X^\top X + n\lambda I)^{-1}w^* - w^* \\ &= -n\lambda(X^\top X + n\lambda I)^{-1}w^*.\end{aligned} \tag{3.12}$$

Notably, the OLS estimator ($\lambda = 0$) remains unbiased:

$$E(\hat{w}_0 - w^*) = 0. \tag{3.13}$$

3.1.3.2 Variance of Ridge Estimator

While introducing bias, ridge regularization simultaneously reduces estimator variance. The variance comparison between OLS and ridge estimators reveals crucial insights. For OLS:

$$\text{Var}(\hat{w}_0) = \sigma^2(X^\top X)^{-1}. \tag{3.14}$$

For ridge regression, we express the variance through a transformed operator:

$$B_\lambda = (X^\top X + n\lambda I)^{-1}X^\top X \tag{3.15}$$

yielding the variance expression:

$$\begin{aligned}
\text{Var}(\hat{w}_\lambda) &= \text{Var}\left[(X^\top X + n\lambda I)^{-1}X^\top Y\right] \\
&= \text{Var}\left[(X^\top X + n\lambda I)^{-1}X^\top X(X^\top X)^{-1}X^\top Y\right] \\
&= \text{Var}\left[B_\lambda \hat{w}_0\right] \\
&= B_\lambda \text{Var}(\hat{w}_0)B_\lambda^\top \\
&= \sigma^2 B_\lambda (X^\top X)^{-1}B_\lambda^\top.
\end{aligned} \tag{3.16}$$

Through eigen-decomposition $X^\top X = U\Sigma U^\top$, we establish:

$$\begin{aligned}
\text{Var}(\hat{w}_0) - \text{Var}(\hat{w}_\lambda) &= \sigma^2(X^\top X)^{-1} - \sigma^2 B_\lambda (X^\top X)^{-1}B_\lambda^\top \\
&= \sigma^2 U\Sigma^{-1}U^\top - \sigma^2(U\Sigma U^\top + n\lambda I)^{-1}U\Sigma U^\top(U\Sigma U^\top + n\lambda I)^{-1} \\
&= \sigma^2 U\Sigma^{-1}U^\top - \sigma^2 U(\Sigma + n\lambda I)^{-1}\Sigma(\Sigma + n\lambda I)^{-1}U^\top \\
&= \sigma^2 U\left[(\Sigma + n\lambda I)^{-1}(\Sigma + 2n\lambda I + n^2\lambda^2\Sigma^{-1} - \Sigma)(\Sigma + n\lambda I)^{-1}\right]U^\top \\
&= \sigma^2(X^\top X + n\lambda I)^{-1}(n^2\lambda^2(X^\top X)^{-1} + 2n\lambda I)(X^\top X + n\lambda I)^{-1}.
\end{aligned} \tag{3.17}$$

By the definition of Eq. 3.11, we have

$$M(\hat{w}_0) - M(\hat{w}_\lambda) = \text{Var}(\hat{w}_0) - \text{Var}(\hat{w}_\lambda) + 0 - (E(\hat{w}_\lambda - w^*))(E(\hat{w}_\lambda - w^*))^\top. \tag{3.18}$$

Next, combining with the bias and variance part, we have:

$$\begin{aligned}
M(\hat{w}_0) - M(\hat{w}_\lambda) &= \sigma^2(X^\top X + n\lambda I)^{-1}(n^2\lambda^2(X^\top X)^{-1} + 2n\lambda I)(X^\top X + n\lambda I)^{-1} \\
&\quad - n^2\lambda^2(X^\top X + n\lambda I)^{-1}(w^* w^{*\top})(X^\top X + n\lambda I)^{-1} \\
&= (X^\top X + n\lambda I)^{-1}\left[\sigma^2(n^2\lambda^2(X^\top X)^{-1} + 2n\lambda I) - n^2\lambda^2 w^* w^{*\top}\right](X^\top X + n\lambda I)^{-1}.
\end{aligned} \tag{3.19}$$

The critical positivity condition simplifies to:

$$2\sigma^2 I - n\lambda w^* w^{*\top} \succeq 0. \tag{3.20}$$

The following lemma provides the decisive criterion:

Lemma 3.1 *For positive definite matrix A and constant C:*

$$C \cdot A - bb^\top \succeq 0 \iff b^\top A^{-1}b \leq C. \tag{3.21}$$

Proof: Via congruence transformation and spectral analysis:

$$\begin{aligned} C \cdot A - bb^\top \succeq 0 &\iff A^{-1/2}(C \cdot A - bb^\top)A^{-1/2} \succeq 0 \\ &\iff C \cdot I - A^{-1/2}bb^\top A^{-1/2} \succeq 0 \\ &\iff \sup_{\|x\|=1} x^\top A^{-1/2}bb^\top A^{-1/2}x \leq C \\ &\iff \|b^\top A^{-\frac{1}{2}}\| \leq \sqrt{c} \\ &\iff b^\top A^{-1}b \leq C. \end{aligned} \tag{3.22}$$

■

Applying Lemma 3.1 yields the MSE dominance condition:

$$\|w^*\|^2 \leq \frac{2\sigma^2}{n\lambda} \iff \lambda \leq \frac{2\sigma^2}{n\|w^*\|^2}. \tag{3.23}$$

This establishes the regularization sweet spot where variance reduction outweighs bias introduction, demonstrating ridge regression's effectiveness in MSE minimization through optimal bias-variance tradeoff.

3.2 Logistic Regression

3.2.1 Differentiate Terms

3.2.1.1 Softmax vs. Softmax for x

The softmaximum of $\mathbf{x}$ is defined as:

$$\ln \sum_{i=1}^{n} e^{x_i} \approx \ln e^{x_{\max}} = x_{\max} \tag{3.24}$$

This approximation holds because the largest term $e^{x_{\max}}$ dominates the sum when exponentials are involved, making the other terms negligible. This approximation is particularly useful when dealing with large values in the exponentials, as it avoids computational overflow.

The softmax function σ takes as input a vector $\mathbf{x}$ of n real numbers and normalizes it into a probability distribution, where the probabilities are proportional to the exponentials of the input numbers. The formula for the softmax function is:

$$\sigma(\mathbf{x}) = \left\{ \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} \right\}_{i=1}^{n} \tag{3.25}$$

which can also be expressed using inner product notation:

$$\sigma(\mathbf{x}) = \frac{\exp(\mathbf{x})}{\langle \exp(\mathbf{x}), \mathbf{1}_n \rangle} \tag{3.26}$$

where $\exp(\mathbf{x})$ is the element-wise exponential of the vector $\mathbf{x}$, and $\mathbf{1}_n$ is the identity vector of size n.

In classification tasks, the softmax function is crucial for converting unnormalized logits (scores) into probabilities. The sum of the outputs of the softmax function always equals 1, making it a valid probability distribution. Specifically, softmax can be viewed as approximating a one-hot encoded vector $\mathbf{y} = (0, 0, \ldots, 1, \ldots, 0, 0)$, where the largest value in the input vector $\mathbf{x}$ is assigned a probability close to 1, while others are assigned values close to 0.

Thus, softmax is widely used in multi-class classification tasks, where each class is represented by a probability corresponding to its logit value, and the model predicts the class with the highest probability.

3.2.2 Cross Entropy

Cross-entropy $H(p, q)$ quantifies the difference between two probability distributions. It represents the expected value of the information content $I_p(x)$ under the distribution q, where p is the predicted distribution and q is the ground truth distribution:

$$H(p, q) = \mathbb{E}_q[I_p(x)] = \mathbb{E}_q[-\log p(x)] = \langle \mathbf{y}, -\log \sigma(\mathbf{x}) \rangle \tag{3.27}$$

This formula captures the cost of predicting the distribution p when the true distribution is q. Since the ground-truth labels y_i are typically one-hot encoded, this simplifies the expression as follows:

$$-\log \prod_{i=1}^{n} [\sigma(\mathbf{x})]^{y_i} \tag{3.28}$$

Here, $\sigma(\mathbf{x})$ is the predicted probability for class i, and y_i is the corresponding true label. In one-hot encoding, only the term corresponding to the correct class contributes, and all other terms vanish.

Cross-entropy is equivalent to the negative log-likelihood. In the context of classification, this means that minimizing cross-entropy is equivalent to maximizing the log-likelihood of the true class (suppose the true class is k):

$$\log \sigma(\mathbf{x})_k \tag{3.29}$$

This representation shows how the cross-entropy loss measures the distance between the predicted probabilities and the true labels. The smaller the cross-entropy, the closer the predicted distribution is to the actual distribution.

In practical machine learning frameworks such as PyTorch, the cross-entropy loss is computed using the unnormalized logits (the raw output of the model before applying softmax). The function `CrossEntropyLoss` in PyTorch combines both the softmax computation and the cross-entropy calculation, so the input to this function should be the raw logits, not probabilities.

3.2.3 Logits and Binary Classification

The logit function is defined as the logarithm of the odds of an event occurring:

$$\log \left(\frac{p}{1-p} \right) \tag{3.30}$$

In binary classification, where we only have two possible outcomes, the probability p of the event occurring is modeled using the sigmoid function σ:

$$p = \sigma(x_1) = \frac{\exp(x_1)}{1 + \exp(x_1)} \tag{3.31}$$

This is the logistic function, which transforms the input x_1 (which can be any real number) into a probability between 0 and 1.

Taking the logit of p, we get:

$$\log \frac{p}{1-p} = \log \exp(x_1) = x_1 \tag{3.32}$$

This shows that the logit is simply the input x_1 itself, which is the natural parameter in the logistic regression model. The logit function is particularly useful in interpreting the coefficients in logistic regression, as it relates directly to the odds of the event occurring.

3.2.4 Alternative Form of Logistic Regression

Logistic regression can also be expressed in an alternative form, where we model the probability of each class i as:

$$\frac{\exp(\mathbf{w}_i^\top \mathbf{x})}{1 + \sum_{j=1}^{n-1} \exp(\mathbf{w}_j^\top \mathbf{x})} \tag{3.33}$$

Here, $\mathbf{w}_n = \mathbf{0}$ represents the n-th class, and this formulation is commonly used in multi-class classification tasks. However, this formulation is overparameterized, meaning that there are redundant parameters in the model.

Overparameterization introduces redundancy in the model parameters, which can lead to numerical instability during optimization. Specifically, when the number of parameters exceeds the necessary amount, it can cause issues with the Hessian matrix, which may become singular (non-invertible). This can lead to difficulties when performing gradient-based optimization, as the optimization algorithm may struggle to converge properly.

When logistic regression has more than two classes, it can be regarded as a generalization of binary logistic regression by using a softmax function to handle multiple classes and probabilities.

3.2.5 Numerical Issues

Numerical issues may occur due to overflow or underflow. For instance:

$$\exp(x_i) \tag{3.34}$$

can lead to overflow when $x_1 = 1000, x_2 = 2000, x_3 = 3000$, as the exponential function grows extremely fast. When the values of x_i are too small, underflow may occur.

To address numerical issues, a more stable form is used:

$$\frac{\exp(x_i - x_{\max})}{\sum \exp(x_j - x_{\max})} \tag{3.35}$$

where $x_{\max} = \max(x_1, \ldots, x_n)$. This form can prevent the computation of unsuitable values in the exponentiation to maintain stability.

The log-softmax function is often preferred over softmax because it improves the numerical stability during gradient backpropagation.

3.2.6 Alternative Cross-Entropy Formulation

In binary classification, where the labels $y \in \{0, 1\}$ or $y \in \{-1, 1\}$, the cross-entropy loss can be rewritten as:

$$e(z) = \log(1 + \exp(-yz)) \tag{3.36}$$

where $\sigma(z) = \frac{\exp(z)}{1+\exp(z)}$ is the sigmoid function, which can rearrange z to [0, 1]. It is equivalent to:

$$P(Y = y|z) = \sigma(yz) \tag{3.37}$$

where z is the linear combination, and $\sigma(yz)$ is the predicted probability. It means the probability of the ground truth y given z.

The negative log-likelihood for this case is given by:

$$e(z) = -\ln \sigma(yz) = \ln(1 + \exp(-yz)) \tag{3.38}$$

3.2.7 Convexity of Logistic Regression

We will explain why logistic regression is considered a convex function. The loss gradient concerning the activation a is:

$$\frac{\partial l}{\partial a} = -y + \text{softmax}(a), \quad a = WX \tag{3.39}$$

Then represent the Hessian matrix H:

$$H = \text{diag}(\sigma) - \sigma\sigma^\top \tag{3.40}$$

Furthermore, compute the i-th column of the Hessian H_i:

$$H_i = \frac{\partial g_i}{\partial a} = \frac{\partial \langle e_i, \frac{\partial l}{\partial a} \rangle}{\partial a} \tag{3.41}$$

Expanding this, we get:

$$\frac{\partial g_i}{\partial a} = \langle e_i, \text{softmax}(a) \rangle [e_i - \text{softmax}(a)]^\top \frac{\partial a}{\partial a} \tag{3.42}$$

which simplifies to:

$$H_i = \sigma_i (e_i - \sigma) \tag{3.43}$$

Thus, the Hessian matrix H can be written as:

$$H = \text{diag}(\sigma) - \sigma\sigma^\top \tag{3.44}$$

For any nonzero vector x, we can calculate the quadratic form $x^\top H x$:

$$x^\top H x = \sum \sigma_i x_i^2 - \left(\sum \sigma_i x_i \right)^2 \geq 0 \tag{3.45}$$

where $\sum \sigma_i = 1$. Since it is always non-negative, H is positive and semi-definite, which implies that the logistic regression loss function is convex.