Loss Functions

Least Sum of Squares (L2 Loss)

$\min_\beta \|X\beta - y\|_2^2$ **Derivation of closed form**

$J(\beta) = \|y - X\beta\|_2^2$
$J(\beta) = (y - X\beta)^T(y - X\beta)$
$J(\beta^*) = \nabla J(\beta) = -2X^T(y - X\beta) = 0$
$X^TX\beta = X^Ty = 0$
$\beta = (X^TX)^{-1}X^Ty$

L-Inf Loss

$L_{1,\infty}(\beta) = \max_g \sum_{i\in g} |y_i - X_i\beta|$ **No closed form solution** outliers pull on this. Use if standard deviation is small.

L1 Loss (Lasso Regression)

$\min_\beta \|X\beta - y\|_2^2 + \lambda\|\beta\|_1$

$L(\beta) = \frac{1}{2n}\sum_{i=1}^n \left(y_i - \sum_{j=1}^d X_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^d |\beta_j|$

No closed-form solution: Requires iterative optimization (e.g., coordinate descent). - **Benefits** Performs feature selection by driving some coefficients to zero, improving interpretability. - **Tradeoff** Excludes features that may still hold important information, may underperform if many features are relevant.

Ridge Regularized L2 Loss

$\|X\beta - y\|_2^2 + \lambda\|\beta\|_2^2$

Derivation of closed form $J(\beta) = \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$
$J(\beta) = (y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta$
$\nabla_\beta J(\beta) = -2X^T(y - X\beta) + 2\lambda\beta = 0$
$X^TX\beta + \lambda\beta = X^Ty$
$(X^TX + \lambda I)\beta = X^Ty$
$\beta = (X^TX + \lambda I)^{-1}X^Ty$ - **Benefits** Regularization avoids overfitting by shrinking coefficients. Prevents overfitting, retains all features. - **Tradeoff** Requires careful tuning of λ for the right balance between bias and variance. Does not perform feature selection, which reduces interpretability in high-dimensional spaces.

Mean Squared Error (MSE)

$\min_\beta \frac{1}{n}\sum_{i=1}^n (y_i - X_i\beta)^2$ - **Closed-form solution** $\beta^* = (X^TX)^{-1}X^Ty$ - **Benefits** Simple and easy to compute. Works well with linear regression. - **Tradeoff** Sensitive to outliers, which can dominate the loss.

Logarithmic Loss (LogLoss)

$L_{\text{logloss}} = -\frac{1}{n}\sum_{i=1}^n [y_i\log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$ **Explanation:** measures the performance of a classification model whose output is a probability value between 0 and 1. It penalizes predictions that are far from the true labels, with larger penalties for confident but incorrect predictions.

Benefits: - Captures the uncertainty of predictions by considering probabilities. - Penalizes incorrect predictions more heavily the more confident they are. - Differentiable, making it suitable for gradient-based optimization.

Tradeoffs: - Sensitive to outliers, especially if the model is highly confident but wrong. - Requires well-calibrated probability outputs to be effective. - Can lead to high loss values if the model is overconfident with incorrect predictions.

Cross-Entropy Loss (Logistic Regression)

$L(\beta) = -\sum_{i=1}^n (y_i\log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i))$ **No closed-form solution** Solved iteratively via gradient descent. - **Explanation** Penalizes incorrect predictions based on the log likelihood of the true class. MLE for β is $\text{argmax}_\beta P(y|\beta) = P(y_1|\beta) * P(y_2|\beta) \cdot P(y_n|beta) = logloss * crossentropyloss$. - **Benefits** Ideal for binary classification, models probabilities effectively. - **Tradeoff** More computationally expensive, requires careful parameter tuning.

Hinge Loss (Support Vector Machines)

$L = \max(0, 1 - y_iX_i\beta)$ **No closed-form solution** Solved via convex optimization methods - **Benefits** Useful in maximizing the margin between classes. - **Tradeoff** Computationally intensive for large datasets.

0-1 Loss (Classification Problems)

$L = \sum_{i=1}^n \nVdash(y_i \neq \hat{y}_i)$, where $\nVdash$ is an indicator function. - **Benefits** Easy to understand and directly penalizes misclassification. - **Tradeoff** Non-convex and discontinuous, so not suitable for gradient-based methods.

Multi-Class Cross-Entropy Loss

- **Objective** Used for multi-class classification problems to measure the performance of a classification model whose output is a probability distribution across multiple classes. - **Formula** $L = -\sum_{i=1}^n\sum_{c=1}^C y_{i,c}\log(\hat{y}_{i,c})$, where C is the number of classes, $y_{i,c}$ is the true label (1 if class c is correct, 0 otherwise), and $\hat{y}_{i,c}$ is the predicted probability for class c. - **Benefits** - Ideal for problems where each instance can belong to one of several classes (e.g., image classification). - Models probabilistic outcomes effectively, providing confidence scores. - **Tradeoffs** - More computationally intensive compared to binary cross-entropy due to multiple classes. - Sensitive to class imbalance, which may lead to biased predictions if one class dominates. - **Key Concepts** This loss encourages models to output probabilities that are as close as possible to the true one-hot encoded labels.

Binary Cross-Entropy Loss

- **Objective** Used for binary classification problems, measuring the performance of a classification model whose output is a probability value between 0 and 1. - **Formula** $L = -\frac{1}{n}\sum_{i=1}^n (y_i\log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i))$, where y_i is the true binary label (1 or 0), and $\hat{y}_i$ is the predicted probability for label 1. - **Benefits** - Ideal for binary classification tasks like spam detection or medical diagnosis. - **Tradeoffs** - Can struggle with class imbalance - Sensitive to extreme predictions (very close to 0 or 1) that may cause large gradients, impacting training stability. - **Key Concepts** This loss penalizes incorrect predictions and emphasizes confidence, making it widely used in classification problems involving two outcomes.

Equivalence of Multi-Class and Binary Cross-Entropy Loss

- **Equivalence** Multi-class cross-entropy simplifies to binary cross-entropy when the number of classes $C = 2$. - **Setup** For binary classification, we set $\beta^{(0)} = -\beta$ and $\beta^{(1)} = \beta$. - **Multi-Class Cross-Entropy** for two classes**:

$L = -\sum_{i=1}^n\sum_{c=0}^1 y_{i,c}\log(\hat{y}_{i,c})$

where $\hat{y}_{i,0} = \sigma(-\beta^Tx_i)$ and $\hat{y}_{i,1} = \sigma(\beta^Tx_i)$. - **Simplification** Plugging in the values of $\hat{y}_{i,0}$ and $\hat{y}_{i,1}$:

$L = -\sum_{i=1}^n \left(y_i\log(\sigma(\beta^Tx_i)) + (1 - y_i)\log(1 - \sigma(\beta^Tx_i))\right)$

This is the Binary Cross-Entropy Loss:

$L = -\frac{1}{n}\sum_{i=1}^n (y_i\log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i))$ - **Conclusion** Multi-class cross-entropy for two classes reduces to binary cross-entropy when $\beta^{(0)} = -\beta$ and $\beta^{(1)} = \beta$.

Gaussian Naive Bayes

- **MAP** $\text{argmax}_y \left(p(y|x) = \frac{p(x|y)p(y)}{p(x)}\right)$ - **Likelihood** $p(x|y) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$ - **Benefits** Fast to compute, assumes independence between features. - **Tradeoff** Assumption of independence is often unrealistic, which can lead to inaccuracies.

Derivatives for Optimization

- **Gradient** $\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]$ - **Chain Rule** $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u}\cdot\frac{\partial u}{\partial x}$

Maximum Likelihood Estimation (MLE)

- **Gaussian MLE** $\mu_{MLE} = \frac{1}{n}\sum x_i, \quad \sigma^2_{MLE} = \frac{1}{n}\sum(x_i - \mu)^2$ - **Bernoulli MLE** $\mu_{MLE} = \frac{1}{n}\sum x_i$ - **Benefits** Provides efficient estimators if the assumptions about data distribution are correct. - **Tradeoff** Assumptions about data distribution can lead to poor results if incorrect.

Gradient Descent

- **Update Rule** $\beta \leftarrow \beta - \alpha\nabla L(\beta)$, where α is the learning rate. - **Benefits** Works for large models without closed-form solutions (e.g., neural networks, logistic regression). - **Tradeoff** Sensitive to choice of learning rate, can converge slowly or diverge.

Rayleigh Distribution MLE

- **PDF** $p(x) = \frac{x}{\sigma^2}\exp\left(\frac{-x^2}{2\sigma^2}\right)$ - **MLE for σ** $\sigma_{MLE} = \sqrt{\frac{1}{2n}\sum x_i^2}$ - **Benefits** Provides a simple estimation method for certain non-negative data. - **Tradeoff** Assumes a specific distribution, may not generalize well to other data.

Matrix Calculus Rules

- **Quadratic Form Derivative** $\frac{d}{d\beta}\left(\beta^TX\beta\right) = 2X\beta$ - **Logarithmic Derivative** $\frac{d}{d\beta}\log f(\beta) = \frac{1}{f(\beta)}\cdot f'(\beta)$

Bias-Variance Tradeoff

- **Benefits** Helps in understanding model complexity, assisting in selecting simpler models to reduce variance or more complex models to reduce bias. - **Tradeoff** High bias leads to underfitting (poor accuracy), high variance leads to overfitting (poor generalization).

Regularization Techniques

L2 Regularization (Ridge)

- **Objective** Adds $\lambda\|\beta\|_2^2$ to the loss function. As lambda grows - **Closed-form solution** $\beta^* = (X^TX + \lambda I)^{-1}X^Ty$ - **Benefits** Prevents overfitting, improves generalizability. - **Tradeoff** Does not eliminate features, making models harder to interpret in high dimensions.

L1 Regularization (Lasso)

- **Objective** Adds $\lambda\|\beta\|_1$ to the loss function. - **No closed-form solution** Solved via optimization (e.g., coordinate descent). - **Benefits** Encourages sparsity, making the model more interpretable. - **Tradeoff** Can exclude relevant features if not tuned carefully.

One-Hot Encoding

- **Definition** One-hot encoding is a process used to convert categorical data into a binary vector for each category. - **Process** Each category in the dataset is transformed into a vector where only one element is 1, and the rest are 0s. - **Benefits** Allows categorical data to be used in machine learning algorithms that require numerical input. - **Tradeoff** Can lead to high-dimensional datasets when the number of categories is large, which may increase computational costs and memory usage.

Distributions

- Laplace **PDF** $p(x) = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)$ - Guassian **PDF** $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ - Bernoulli **PMF** $p(x) = \mu^x(1-\mu)^{1-x}$ - Rayleigh **PDF** $p(x) = \frac{x}{\sigma^2}\exp\left(-\frac{x^2}{2\sigma^2}\right)$

Empirical Risk Minimization (ERM) and Population Risk

- **ERM** Minimize loss over the training dataset. Objective: $\hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i), y_i)$. - **Population Risk** The expected loss over the entire distribution: $L(f) = \mathbb{E}_{x,y}[\ell(f(x), y)]$. - **Key Concept** In practice, we minimize ERM as the true distribution is unknown.

Logistic Regression - Gradient and Hessian

- **Objective** Minimize cross-entropy loss. For binary classification:

$L(\beta) = -\sum_{i=1}^{n}(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i))$

- **Gradient**

$\nabla_\beta L(\beta) = \sum_{i=1}^{n}(\hat{y}_i - y_i)\,x_i$

$\nabla_\beta L(\beta) = X^T(\hat{y} - y) = 0$

$h(z) = \frac{1}{1+e^{-z}} = sigmoid$ - **Hessian** The second derivative (useful in Newton's method for optimization). $H = \sum_{i=1}^{n}\hat{y}_i(1-\hat{y}_i)x_i x_i^T$

$H = X^T W X$, where W is a diagonal matrix with elements $\hat{y}_i(1-\hat{y}_i)$.

- **Key Insight** Logistic regression is best suited for linearly separable data. The Hessian matrix H is used in second-order optimization, such as Newton's method, to refine parameter estimates. It provides the second derivatives of the log-likelihood, helping adjust parameter updates based on the curvature of the loss function. Newton's method uses the Hessian in the update rule:

$\beta_{t+1} = \beta_t - H^{-1}\nabla L(\beta)$

Bias-Variance Decomposition

- **Total Error** Decomposed into bias, variance, and irreducible error.

Error = Bias^2 + Variance + Irreducible Error - **High Bias** Underfitting, model too simple. - **High Variance** Overfitting, model too complex. - **Tradeoff** More complex models may have lower bias but higher variance.

Polynomial Regression - Transforming Data

- **Objective** Linear regression on transformed features (polynomial terms).
- **Key Idea** A linear model is applied to a polynomial transformation of the data:

$$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d$$

- **Use Case** Fits data better when non-linear relationships are present.

L1 vs L2 Regularization

- **L1 (Lasso)** – Performs feature selection by driving some coefficients to zero. – Useful when some features are irrelevant. - **L2 (Ridge)** – Shrinks coefficients but retains all features. – Better when most features contribute to the prediction. - **Elastic Net** Combination of L1 and L2, useful when some features should be excluded, but correlation between features exists.

One-vs-One vs One-vs-All for Multi-Class

** - **One-vs-All** Train one classifier per class (class vs all others). Suited for imbalanced data. - **One-vs-One** Train classifiers between each pair of classes. Scales better for many classes but computationally intensive.

0-1 Loss - Computational Intractability

** - **Objective** $\ell_{0-1} = \nVdash(y_i \neq \hat{y}_i)$. - **Key Concept** Minimizing 0-1 loss directly is NP-hard, which is why surrogates like hinge or logistic loss are used.

Newton's Method for Logistic Regression

** - **Objective** Update weights using second-order information. - **Update Rule**

$$\beta_{t+1} = \beta_t - H^{-1}\nabla L(\beta)$$

- **Use Case** Logistic regression, where the Hessian matrix speeds up convergence.

k-Fold Cross Validation

** - **Key Concept** Split data into k subsets. Train on k-1, test on 1. Repeat k times, each subset as test. Average results for final performance estimate.
- **Use Case** Ensures that each data point is used for both training and validation. - **Tradeoff** Computationally expensive but gives better estimates of model performance.

Recall, Precision, AUC

- **Recall**: $\frac{\text{TP}}{\text{TP+FN}}$. As λ goes up, recall goes down. - **Precision**: $\frac{\text{TP}}{\text{TP+FP}}$. As λ goes up, precision goes up.

Other Concepts To Remember

Poor Fit in Linear Regression

Poor fitting in linear regression can occur when the wrong model is chosen. For example, non-linear data can be handled by transforming the target variable y, such as solving for β with respect to $1/y$. This transformation allows for better fitting of non-linear relationships.

Multiple Linear Regression and Data Transformations

Multiple linear regression can be adapted to fit more complex models, such as q-degree polynomials or sinusoidal curves, by transforming the input features. For example, creating a transformation vector with polynomial or sinusoidal features allows the use of techniques like LASSO to select the most relevant features. Sinusoidal transformations are particularly useful for periodic data.

Overfitting and Generalization

Overfitting occurs when the model performs well on training data but poorly on test data, indicating poor generalization. Generalization refers to the model's ability to predict accurately on unseen data. A proper split of the dataset, where the training set is significantly larger than the test set, can help assess generalization.

Model Selection

Model selection involves choosing the appropriate model to represent the data. The right model helps balance the fit on training data and test data to avoid overfitting or underfitting.

Risk Definitions

- **Population Risk**: The average loss over the entire dataset. - **Empirical Risk**: The average loss over the training data. Since the expectations of the population and empirical risks are the same, empirical risk serves as a good estimate of generalization performance.

Regularization

Regularization adds a penalty to the loss function to prevent overfitting, especially when there are more features than data points, which can lead to a model that fits the training data too closely. Regularization reduces the values of β_*, thereby controlling the model's complexity.

Bayesian Perspective of Regularization

Regularization can be justified from a Bayesian perspective, where it serves as a prior belief about the parameters. By incorporating a prior, we reduce overfitting by shrinking the parameter estimates toward more likely values.

MAP vs. MLE

- **Maximum A Posteriori (MAP)**: Estimates the parameters by maximizing the posterior distribution, considering both prior knowledge and the likelihood of the data. - **Maximum Likelihood Estimation (MLE)**: Estimates the parameters by maximizing the likelihood function without considering prior information.

Naive Bayes Classifier and Binary Vectors

The Naive Bayes Classifier is justified by the Bernoulli probability model for binary vectors. The model's parameters are learned from the frequency of occurrences in the past data.

Other Bayesian Concepts

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

$p(x, y) = p(x|y)p(y)$

$P(x|y) < p(x, y)$

More Stuff

$\nabla f(x) = e^{<a,x>} = ae^{<a,x>}$

Rules for Derivatives

Power Rule: $\frac{d}{dx}[x^n] = nx^{n-1}$ **Constant Rule:** $\frac{d}{dx}[c] = 0$ **Constant Multiple Rule:** $\frac{d}{dx}[cf(x)] = c\frac{d}{dx}[f(x)]$ **Sum Rule:** $\frac{d}{dx}[f(x) + g(x)] = \frac{d}{dx}[f(x)] + \frac{d}{dx}[g(x)]$ **Product Rule:** $\frac{d}{dx}[f(x)g(x)] = f'(x)g(x) + f(x)g'(x)$ **Quotient Rule:** $\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right] = \frac{f'(x)g(x)-f(x)g'(x)}{[g(x)]^2}$ **Chain Rule:** $\frac{d}{dx}[f(g(x))] = f'(g(x)) \cdot g'(x)$ **Exponential Rule:** $\frac{d}{dx}[e^x] = e^x$ **Logarithmic Rule:** $\frac{d}{dx}[\log(x)] = \frac{1}{x}$ **1Trigonometric Rules:** $\frac{d}{dx}[\sin(x)] = \cos(x)$ $\frac{d}{dx}[\cos(x)] = -\sin(x)$ $\frac{d}{dx}[\tan(x)] = \sec^2(x)$ **Inverse Trigonometric Rules:** $\frac{d}{dx}[\sin^{-1}(x)] = \frac{1}{\sqrt{1-x^2}}$ $\frac{d}{dx}[\cos^{-1}(x)] = -\frac{1}{\sqrt{1-x^2}}$ $\frac{d}{dx}[\tan^{-1}(x)] = \frac{1}{1+x^2}$ **Natural Logarithm Rule:** $\frac{d}{dx}[\ln(x)] = \frac{1}{x}$