1. L1 and L2 Loss Functions with Benefits and Tradeoffs

L2 Loss (Ridge Regression)

- **Objective**: $\min_\beta \|X\beta - y\|_2^2 + \lambda\|\beta\|_2^2$ - **Closed-form solution**: $\beta^* = (X^TX + \lambda I)^{-1}X^Ty$ - **Benefits**: Shrinks coefficients, prevents overfitting, retains all features. - **Tradeoff**: Does not perform feature selection, which reduces interpretability in high-dimensional spaces.

L1 Loss (Lasso Regression)

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

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

- **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.

2. Ridge Regularized Least Squares

- **Objective**: $\|X\beta - y\|_2^2 + \lambda\|\beta\|_2^2$ - **Closed-form solution**: $\beta^* = (X^TX + \lambda I)^{-1}X^Ty$ - **Benefits**: Regularization avoids overfitting by shrinking coefficients. - **Tradeoff**: Requires careful tuning of λ for the right balance between bias and variance.

3. Loss Functions

Mean Squared Error (MSE)

- **Objective**: $\min_\beta \frac{1}{n}\sum_{i=1}^{n}(y_i - X_i\beta)^2$ - **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.

Cross-Entropy Loss (Logistic Regression)

- **Objective**: $L(\beta) = -\sum_{i=1}^{n}(y_i\log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i))$ - **No closed-form solution**: Solved iteratively via gradient descent. - **Benefits**: Ideal for binary classification, models probabilities effectively. - **Tradeoff**: More computationally expensive, requires careful parameter tuning.

Hinge Loss (Support Vector Machines)

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

0-1 Loss (Classification Problems)

- **Objective**: $L = \sum_{i=1}^{n}\not{\mathbb{K}}(y_i \neq \hat{y}_i)$, where $\not{\mathbb{K}}$ 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 $\hat{y}_{i,1}$:

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

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 $\beta^{(1)} = \beta$.

4. 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.

5. K-Fold Cross Validation

- **Benefits**: Provides better estimates of model performance by using every data point for both training and validation. - **Tradeoff**: Computationally expensive, especially for large datasets or complex models.

6. 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**: $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u}\cdot\frac{\partial u}{\partial x}$

7. 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.

8. 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.

9. 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.

10. Matrix Calculus Rules

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

11. 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).

12. Regularization Techniques

L2 Regularization (Ridge)

- **Objective**: Adds $\lambda\|\beta\|_2^2$ to the loss function. - **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.

13. 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.

14. 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**: $p(x) = \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right)$

1. 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.

2. 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**:

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

- **Hessian**: The second derivative (useful in Newton's method for optimization).

$$H = \sum_{i=1}^{n} \hat{y}_i(1-\hat{y}_i) x_i x_i^T$$

- **Key Insight**: Logistic regression is best suited for linearly separable data.

3. Bias-Variance Decomposition

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

$$\text{Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

- **High Bias**: Underfitting, model too simple. - **High Variance**: Overfitting, model too complex. - **Tradeoff**: More complex models may have lower bias but higher variance.

4. 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.

5. 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.

6. 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.

7. Learning Rate in Gradient Descent

** - **Low Learning Rate**: Convergence is slow but safe. - **High Learning Rate**: Faster convergence but risk of overshooting. - **Optimal Choice**: Adaptive methods like Adam adjust learning rates.

8. 0-1 Loss - Computational Intractability

** - **Objective**: $\ell_{0-1} = \not\Vdash(y_i \neq \hat{y}_i)$. - **Key Concept**: Minimizing 0-1 loss directly is NP-hard, which is why surrogates like hinge or logistic loss are used.

9. 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.

10. k-Fold Cross Validation

** - **Key Concept**: The dataset is split into k subsets, with each subset used once as the validation set and k-1 subsets used for training. - **Use Case**: Ensures that each data point is used for both training and validation. - **Tradeoff**: Computationally expensive but gives better estimates of model performance.

11. Gaussian Naive Bayes - Why Independence Assumption?

- **Key Assumption Features are conditionally independent given the class label. - **Benefit Allows efficient computation of probabilities in high-dimensional spaces. - **Tradeoff: The assumption is rarely true in practice, but works well for tasks like text classification (e.g., spam detection).

12. ROC Curves in Binary Classification

- ** Recall = True Positive Rate (TPR): $\frac{\text{TP}}{\text{TP+FN}}$. - ** Precision = False Positive Rate (FPR): $\frac{\text{FP}}{\text{FP+TN}}$. - ** Area Under the Curve (AUC) **: Measures classifier performance; AUC close to 1 indicates a good classifier.