1 Loss functions

fit works, primarily in the context of linear models. A loss function is a mathematical function that quantifies how well a machine learning model's predictions align with the actual target values. It measures the "cost" associated with prediction errors. During training the model tries to improve its performance by minimizing this cost or loss function.

1.1 ML Training

1. Choose your model (eg. Linear Regression)
2. Choose your loss function (eg. Ordinary Least Squeares)
3. Choose your optimization algorithm (eg. Gradient Descent)

2 Regularization

Complex models often overfit the training data. For example, in polynomial regression, the validation score usually increases at first (as complexity helps) but eventually decreases (as the model becomes too flexible). Real-world relationships between X and Y may be complex, but we rarely know which features truly matter, so we need tools to control model complexity.

2.1 Controlling model complexity

Common ways to control complexity include:

- **Feature selection:** Reduce the number of input features.
- **Ensemble averaging:** Combine multiple models (e.g., random forests) to reduce variance.
- **Regularization:** Add a penalty to the loss function that increases with model complexity.

With regularization, we minimize:

$$\text{Loss} + \lambda\,(\text{Model Complexity})$$

The Loss measures fit to data, while the regularization term penalizes complexity. The parameter λ controls how strong the penalty is.
L0 regularization: Counts the number of non-zero weights:

$$\|\mathbf{w}\|_0 = \#\{w_i : w_i \neq 0\}$$

L1 regularization: Sums absolute values of the weights:

$$\|\mathbf{w}\|_1 = \sum_i |w_i|$$

Makes one of the weights to zero for multi-colinear features
L2 regularization: Sums squared weights:

$$\|\mathbf{w}\|_2^2 = \sum_i w_i^2$$

Gives equal but reduce weightage to multi-colinear features
Idea: L0: penalizes the number of features. (feature selection RFE) L1: encourages sparsity (many zeros). (Lasso Regression) L2: encourages small, smooth weights. (Ridge Regression)

2.2 Why are small weights better?

Somewhat non-intuitive. Suppose x_1 and x_2 are nearby each other. We might expect that they have similar y. If we change feature1 value by a small amount, leaving everything else the same, we might think that the prediction would be the same. But if we have bigger weights, small change has a large effect on the prediction.
Ridge: Linear Regression with L2 regularization Logistic Regression with L2 regularization

3 Ensemble Learning

Group of models or predictors. Often get better and reliable predictions than with a single best model. **High Variance:** high error during testing and validation. Overfitting. **High bias:** high error during training. Underfitting.

4 Ensemble techniques

Greater the diversity among the individual models, the better the ensemble will perform compared to its individual model. This is because different models capture different patterns in the data, and by combining their predictions, the ensemble can leverage the strengths of each model while mitigating their weaknesses.

4.1 Bagging (Bootstrap Aggregating):Random Forests

Several models are trained independently (can be done in parallel) on different random subsets of the data and their predictions are averaged (for regression) or voted upon (for classification).

4.2 Boosting:Gradient Boosted Trees (e.g., XGBoost, LightGBM, CatBoost)

Weak learner are trained sequentially where each new model focuses on correcting the errors made by the previous models. The final prediction is a weighted combination of all models.

4.3 Stacking (stacked generalization):SuperLeraner (R package)

Base learner models of different types (different algorithm) are trained on the same dataset and a meta-model is trained to combine their predictions. Stacking is a heterogenous parallel method.