Machine Learning Methods

Topic 2: Ensembles, Linear Models & Regularization

Yonghao Lee

January 2026

Contents

1 The Difficulty of Finding Optimal ERM Ensembles

Effective ensembles rely on the principle of **diversity**. If all models in an ensemble make identical predictions, the voting mechanism is meaningless; it is essentially multiple copies of the same hypothesis. To achieve a significant reduction in error, the individual learners must be **uncorrelated**—meaning they should make errors on *different* training examples.

Definition 1.1: Ensemble ERM Problem

We seek a set of p shallow trees $\{h_1, \dots, h_p\} \in \mathcal{H}$ that minimize the average classification error over n samples. The formal objective is:

$$h_1^*, \dots, h_p^* = \arg \min_{h_1,\dots,h_p \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\left(y_i \sum_{k=1}^{p} h_k(x_i) < 0\right)}$$

Understanding the Formula

- **Additive Prediction:** The term $\sum_{k=1}^{p} h_k(x_i)$ represents the aggregate vote of the ensemble for sample x_i.
- **Mistake Condition:** The term $y_i(\dots) < 0$ checks for a sign mismatch.
  - If $y_i = +1$ and the sum is negative, the product is negative (Mistake).
  - If $y_i = -1$ and the sum is positive, the product is negative (Mistake).
- **Indicator Function ($\mathbb{1}$):** This captures the 0/1 loss. It returns 1 if the condition (mistake) is true, and 0 if the ensemble is correct.

This optimization is considered hard because it is a **joint optimization**. Unlike training a single tree, you cannot simply pick the p best trees independently. An optimal team might include a tree that is individually weak (has high error on its own) but happens to correctly classify specific difficult samples that all other $p-1$ trees miss.

Study Note

Combinatorial Explosion: To find the absolute minimum empirical risk, one would theoretically need to enumerate every possible subset of p hypotheses from the space $\mathcal{H}$. If $\mathcal{H}$ contains N candidate trees, the number of combinations to check is:

$$\binom{N}{p} = \frac{N!}{p!(N-p)!}$$

As N grows, this search becomes computationally intractable. This is why we cannot solve for the perfect ensemble directly.

1.1 Boosting

Since the joint optimization is impossible, we use **Boosting**, which is a **greedy approximation** algorithm. Instead of finding p models simultaneously, it constructs the ensemble iteratively (one by one).

Definition 1.2: Greedy Boosting Step

At step p, we fix the previous ensemble (models 1 to $p-1$) and find a new model h_p that minimizes risk when added to the group:

$$h_p^* = \arg \min_{h_p \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\left(y_i \left[\sum_{k=1}^{p-1} h_k^*(x_i) + h_p(x_i)\right] < 0\right)}$$

- **Greedy Property:** The previous models $\{h_1^*, \dots, h_{p-1}^*\}$ are *fixed*. The algorithm does not look back to update them; it only optimizes for the current addition, h_p.

- **Residual Focus:** In practice, h_p is incentivized to correctly classify the samples where the existing sum $\sum_{k=1}^{p-1} h_k^*(x_i)$ is currently wrong or weak.
- **Efficiency:** This reduces a massive combinatorial search into p separate searches, making the training of large ensembles feasible.

2 Linear Hypothesis Classes

Decision trees are inherently limited by their **inductive bias**: they split data using axis-aligned hyperplanes. While effective for some datasets, they struggle with patterns that are not axis-aligned. To address this, we introduce **linear decision rules**, which can separate data at any angle.

Definition 2.1: Linear Hypothesis

A linear classifier defines a separating hyperplane. In a 2D feature space, this represents a line splitting a quadrant into two sections:

$$\mathbf{w} \cdot \mathbf{x} + b > 0$$

2.1 The Notation Trick (Augmentation)

To simplify optimization, we avoid tracking the weight vector $\mathbf{w}$ and bias b separately. We absorb the bias into the weight vector by augmenting our vectors:

- $\mathbf{x}_{\text{new}} = [1, x_1, x_2, \ldots, x_d]^\top$
- $\mathbf{w}_{\text{new}} = [b, w_1, w_2, \ldots, w_d]^\top$

This allows us to express the decision rule as a single dot product: $\mathbf{w}_{\text{new}} \cdot \mathbf{x}_{\text{new}} > 0$.

Example 2.1: Pneumonia Diagnosis

Consider a patient being diagnosed based on temperature (t) and cough rate (c).

- **Tree approach (Axis-Aligned):** Requires multiple splits (e.g., If $t > 38$ then If $c > 10$) to approximate a diagonal boundary. This results in a staircase boundary.
- **Linear approach:** Combines symptoms into a single weighted score:

$$f(x) = w_0 + w_1 t + w_2 c > 0$$

If w_1 and w_2 are both positive, the model captures the **synergy** between symptoms: a patient with a moderate fever AND a moderate cough might be classified as sick, even if neither symptom alone crosses the strict thresholds of the decision tree.

2.2 Learning Linear Models

Training linear models presents two primary challenges:

1. **Generalization:** Can we learn from finite samples?
2. **Efficiency:** How do we search for the ERM among infinite models?

Theorem 2.1: VC Dimension of Linear Models

For linear models in d dimensions, the VC dimension is exactly $d + 1$.

- **Inductive Bias:** This finite VC dimension provides a strong inductive bias, preventing the model from perfectly fitting (overfitting) arbitrary noise.
- **Expressivity:** While limited to linear separations, they are robust and highly interpretable.

To address the second challenge (Efficiency), we rely on **convex optimization**. A function $L(\mathbf{w})$ is convex if the line segment between any two points on the graph lies above the graph ($\nabla^2 L(\mathbf{w}) \succeq 0$).

Definition 2.2: Gradient Descent Update

Since the loss function L is convex, any local minimum is a global minimum. We find it using the update rule:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla_{\mathbf{w}} L(\mathbf{w}_t)$$

where $\eta > 0$ is the **learning rate** (step size).

3 Regression

So far, we measured risk using accuracy (0/1 loss). However, the gradient of the 0/1 loss is zero almost everywhere (and undefined at the jump). This means gradient descent cannot move from the initial weight $\mathbf{w}$, as there is no signal to indicate the direction of improvement.

To utilize gradient-based optimization, we introduce **Regression**. Here, the goal is to predict a continuous scalar output (e.g., predicting life expectancy from GDP) rather than a discrete label.

Definition 3.1: Squared Loss (Empirical Risk)

The standard loss for regression is the Squared Loss. The empirical risk over the training set S_{train} is:

$$L(S_{\text{train}}, \mathbf{w}) = \frac{1}{n} \sum_{(\mathbf{x},y) \in S_{\text{train}}} (\mathbf{w} \cdot \mathbf{x} - y)^2$$

Finding the ERM model involves minimizing this convex objective:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} L(S_{\text{train}}, \mathbf{w})$$

3.1 Gradient Descent for Regression

To find $\mathbf{w}^*$, we compute the gradient of the loss with respect to $\mathbf{w}$.

Theorem 3.1: Gradient of Squared Loss

Using the chain rule, the gradient is:

$$\nabla_{\mathbf{w}} L(S_{\text{train}}, \mathbf{w}) = \frac{2}{n} \sum_{(\mathbf{x},y) \in S_{\text{train}}} \mathbf{x}(\mathbf{w} \cdot \mathbf{x} - y)$$

At each iteration t, we update the model to find a lower empirical risk using the formula:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \left[\frac{2}{n} \sum_{i=1}^{n} \mathbf{x}_i (f_{\mathbf{w}_t}(\mathbf{x}_i) - y_i) \right]$$

4 L2 Regularization (Ridge Regression)

We often encounter the problem of **overfitting**, where a model learns the noise in the training data rather than the underlying pattern. This usually manifests as a model with very large weights ($\|\mathbf{w}\|$ is large), creating a complex, unstable function.

To prevent this, we measure complexity by the squared magnitude of the weights, $\|\mathbf{w}\|_2^2$. In practice, we add this penalty term to our loss function. This technique is known as **L2 Regularization** or **Ridge Regression**.

Definition 4.1: Ridge Regression Loss

We modify the standard regression loss by adding the regularization term, scaled by a hyperparameter α:

$$L(S_{train}, \mathbf{w}) = \underbrace{\frac{1}{2N_{train}} \sum_{(\mathbf{x},y)\in S_{train}} (y - \mathbf{w}\cdot\mathbf{x})^2}_{\text{Data Fidelity}} + \underbrace{\frac{\alpha}{2}\|\mathbf{w}\|_2^2}_{\text{Complexity Penalty}}$$

- If $\alpha = 0$: No regularization (standard Linear Regression).
- If $\alpha \to \infty$: The penalty dominates, forcing $\mathbf{w} \to \mathbf{0}$ (the model ignores the data).

4.1 Gradient Descent with Regularization (Weight Decay)

To see how this helps, let us look at the gradient. The gradient of the regularization term $\frac{\alpha}{2}\|\mathbf{w}\|^2$ is simply $\alpha\mathbf{w}$.

$$\nabla L(\mathbf{w}) = \underbrace{\frac{1}{N_{train}} \sum (\dots)}_{\text{Original Gradient } \mathbf{g}} + \alpha\mathbf{w}$$

If we use Gradient Descent with a learning rate η, the update rule becomes:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\mathbf{g} + \alpha\mathbf{w}_t)$$

Rearranging the terms reveals the intuition:

$$\mathbf{w}_{t+1} = \underbrace{(1 - \eta\alpha)\mathbf{w}_t}_{\text{Weight Decay}} - \eta\mathbf{g}$$

4.2 Analytical Solution

Unlike many other models (like Neural Networks), Ridge Regression has a closed-form analytical solution.

Theorem 4.1: Ridge Analytical Solution

By setting the gradient to zero and solving for $\mathbf{w}$, we obtain:

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X} + N_{train}\alpha\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

where $\mathbf{I}$ is the identity matrix. The addition of the term $\alpha\mathbf{I}$ ensures that the matrix is invertible, effectively solving the issue of collinear features.