Lecture 4: Decision theory and Model selection

谢丹
清华大学数学系

October 22, 2025

CHAPTER 5:
Decision theory and Model selection

Section 1: decision theory

The Fundamental Question of ML

Machine learning models produce predictions, but these predictions are inherently uncertain.

Given this uncertainty, what is the best possible action to take?

Decision Theory provides the rigorous mathematical framework to answer this question by formally incorporating loss functions and probabilities.

Core Components of a Decision Problem

A decision-theoretic problem in ML is defined by a 4-tuple $(\mathcal{Y}, \mathcal{A}, P, L)$:

- $\mathcal{Y}$: **State Space** - The set of all possible true states of the world y.
- $\mathcal{A}$: **Action Space** - The set of all possible actions (decisions/predictions) a we can take.
- $P(y|\boldsymbol{x})$: **Uncertainty Model** - A conditional probability distribution over states y given observed data $\boldsymbol{x}$. This is what our ML model provides.
- $L(y, a)$: **Loss Function** - A function quantifying the cost of taking action a when the true state is y.

From Loss to Expected Loss (Risk)

We cannot minimize the loss $L(y, a)$ directly because y is unknown.

Instead, we minimize its expectation over the possible states y, weighted by their probability. This is the **Expected Loss** or **Risk**.

Conditional Risk (Given data $\boldsymbol{x}$)

For a specific input $\boldsymbol{x}$, the risk of an action a is:

$$R(a|\boldsymbol{x}) = \mathbb{E}_{y \sim P(y|\boldsymbol{x})}\left[L(y, a)\right] = \begin{cases} \sum_{y \in \mathcal{Y}} L(y, a)\, P(y|\boldsymbol{x}) & \text{(Discrete)} \\ \int_{\mathcal{Y}} L(y, a)\, p(y|\boldsymbol{x})\, dy & \text{(Continuous)} \end{cases}$$

The Optimal Decision Rule

The optimal action a^* is the one that minimizes this conditional risk.

Bayes Decision Rule

$$a^* = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \, R(a|\boldsymbol{x}) = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \, \mathbb{E}_{y \sim P(y|\boldsymbol{x})} \left[L(y, a) \right]$$

- This decision rule is optimal in the sense that it minimizes the total expected loss for the given model $P(y|\boldsymbol{x})$.
- The minimal achievable risk, $R(a^*|\boldsymbol{x})$, is called the **Bayes Risk**.

Case 1: 0-1 Loss (Classification)

0-1 Loss Function:

$$L(y, a) = \mathbb{I}(y \neq a) = \begin{cases} 0 & \text{if } y = a \\ 1 & \text{if } y \neq a \end{cases}$$

Risk for a specific action $a = j$:

$$\begin{aligned} R(a = j|\boldsymbol{x}) &= \sum_{y \neq j} 1 \cdot P(y|\boldsymbol{x}) + 0 \cdot P(y = j|\boldsymbol{x}) \\ &= \sum_{y \neq j} P(y|\boldsymbol{x}) = 1 - P(y = j|\boldsymbol{x}) \end{aligned}$$

Optimal Decision: Maximum A Posteriori (MAP)

To minimize the risk $1 - P(y = j|\boldsymbol{x})$, we maximize $P(y = j|\boldsymbol{x})$.

$$a^* = \underset{j \in \mathcal{Y}}{\operatorname{argmax}} \, P(y = j|\boldsymbol{x})$$

Case 2: Squared Error Loss (Regression)

Squared Error Loss:

$$L(y, a) = (y - a)^2$$

Conditional Risk:

$$R(a|\boldsymbol{x}) = \mathbb{E}_{y \sim p(y|\boldsymbol{x})} \left[(y - a)^2 \right]$$

Optimal Decision: Mean of the Posterior

We find a^* by taking the derivative of the risk and setting it to zero:

$$\begin{aligned} \frac{\partial R(a|\boldsymbol{x})}{\partial a} &= \frac{\partial}{\partial a} \mathbb{E}[(y - a)^2] = \mathbb{E}\left[\frac{\partial}{\partial a} (y - a)^2 \right] = \mathbb{E}[-2(y - a)] \\ &= -2(\mathbb{E}[y|\boldsymbol{x}] - a) \stackrel{!}{=} 0 \\ &\Rightarrow a^* = \mathbb{E}[y|\boldsymbol{x}] = \int y \, p(y|\boldsymbol{x}) \, dy \end{aligned}$$

Case 3: Absolute Error Loss (Regression)

Absolute Error Loss:

$$L(y, a) = |y - a|$$

Conditional Risk:

$$R(a|\boldsymbol{x}) = \mathbb{E}_{y \sim p(y|\boldsymbol{x})} \left[|y - a| \right] = \int |y - a| \, p(y|\boldsymbol{x}) \, dy$$

Optimal Decision: Median of the Posterior

The minimizer a^* of the expected absolute error is the **median** of the distribution $p(y|\boldsymbol{x})$, satisfying:

$$\int_{-\infty}^{a^*} p(y|\boldsymbol{x}) \, dy = \int_{a^*}^{\infty} p(y|\boldsymbol{x}) \, dy = 0.5$$

(Proof involves taking the subderivative of the risk function).

A Principled Workflow

Decision theory fits naturally within the Bayesian framework:

1. **Prior:** Start with prior belief $P(y)$.
2. **Data:** Observe data $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}$.
3. **Posterior:** Compute the posterior distribution using Bayes' Theorem:

$$P(y|\mathcal{D}) \propto P(\mathcal{D}|y)\, P(y)$$

   This posterior encapsulates all our uncertainty about y.
4. **Loss:** Define a loss function $L(y, a)$ relevant to the task.
5. **Decision:** Choose the optimal action that minimizes the posterior expected loss:

$$a^* = \underset{a \in \mathcal{A}}{\operatorname{argmin}}\, \mathbb{E}_{y \sim P(y|\mathcal{D})} \left[L(y, a) \right]$$

Asymmetric and Composite Loss Functions I

Real-world costs are often not symmetric.

Example

Medical Diagnosis

- **False Negative (FN):** $L(\text{cancer}, \text{healthy}) = C_{\text{FN}}$ (Very high cost)
- **False Positive (FP):** $L(\text{healthy}, \text{cancer}) = C_{\text{FP}}$ (Lower cost)

We can define a cost matrix:

	Predict Healthy (a_0)	Predict Cancer (a_1)
True Healthy (y_0)	0	C_{FP}
True Cancer (y_1)	C_{FN}	0

The optimal decision rule becomes: Predict Cancer (a_1) if

$$P(y_1|\boldsymbol{x}) \cdot C_{\text{FN}} > P(y_0|\boldsymbol{x}) \cdot C_{\text{FP}} \quad \text{or} \quad P(y_1|\boldsymbol{x}) > \frac{C_{\text{FP}}}{C_{\text{FP}} + C_{\text{FN}}}$$

This changes the decision threshold from 0.5.

Rejection Option

Sometimes the optimal decision is to make no decision.

Introduce a reject action a_R with a constant cost λ_R.

Modified Decision Rule

Choose class j if $R(a = j|\boldsymbol{x}) < R(a = k|\boldsymbol{x})\ \forall k \neq j$ **and**
$R(a = j|\boldsymbol{x}) < \lambda_R$
Otherwise, reject.

For 0-1 loss, this simplifies to:

- Choose class j if $P(y = j|\boldsymbol{x}) > \max_{k \neq j} P(y = k|\boldsymbol{x})$ **and** $P(y = j|\boldsymbol{x}) > 1 - \lambda_R$
- Otherwise, reject.

This avoids making predictions when the model is highly uncertain.

Summary

- Decision Theory provides the mathematical basis for turning probabilistic predictions ("what we believe") into optimal actions ("what we do").
- The core idea is to minimize the expected loss (risk).
- The choice of loss function determines the optimal decision:
  - 0-1 Loss $\rightarrow$ Mode (MAP)
  - Squared Loss $\rightarrow$ Mean
  - Absolute Loss $\rightarrow$ Median
- It is crucial for applications with asymmetric costs and allows for sophisticated strategies like rejection.

Section 2: Model selection: bias-variance trade-off

The Fundamental Goal: Generalization

We don't just want a model that performs well on the data it was trained on (training error).

We want a model that performs well on new, unseen data (generalization error or test error). Decision theory gives a way to

choose the best action for a given model. If we have many models, how to choose the model?

Two Paths to Failure

Underfitting (High Bias)

- Model is too simple.
- Fails to capture patterns.
- High error on training data.

Overfitting (High Variance)

- Model is too complex.
- Captures noise as if it were pattern.
- Low error on training data, high error on test data.

The True Model and Our Approximation

Assume a true underlying relationship between input x and output y:

$$y = f(x) + \epsilon$$

where:

- $f(x)$ is the deterministic true function.
- ϵ is random noise with $\mathbb{E}[\epsilon] = 0$ and $\mathrm{Var}(\epsilon) = \sigma_\epsilon^2$.

Our goal is to learn an estimator $\widehat{f}(x)$ from a finite training dataset $\mathcal{D}$ to approximate $f(x)$.

Defining the Error

For a fixed test point x, the expected prediction error is the average error if we repeated the model building process over many different training sets $\mathcal{D}$:

$$\text{Error}(x) = \mathbb{E}_{\mathcal{D}}\left[(y - \widehat{f}_{\mathcal{D}}(x))^2\right]$$

Note: The expectation $\mathbb{E}_{\mathcal{D}}$ is over all possible training datasets.

Step 1: Expand the Square

$$
\begin{aligned}
\mathbb{E}_{\mathcal{D}}\left[(y-\widehat{f})^2\right] &= \mathbb{E}_{\mathcal{D}}\left[(f+\epsilon-\widehat{f})^2\right] \\
&= \mathbb{E}_{\mathcal{D}}\left[\Big((f-\mathbb{E}[\widehat{f}])+(\mathbb{E}[\widehat{f}]-\widehat{f})+\epsilon\Big)^2\right] \\
&= \mathbb{E}_{\mathcal{D}}\Big[(f-\mathbb{E}[\widehat{f}])^2+(\mathbb{E}[\widehat{f}]-\widehat{f})^2+\epsilon^2 \\
&\quad +2(f-\mathbb{E}[\widehat{f}])(\mathbb{E}[\widehat{f}]-\widehat{f})+2(f-\mathbb{E}[\widehat{f}])\epsilon+2(\mathbb{E}[\widehat{f}]-\widehat{f})\epsilon\Big]
\end{aligned}
$$

Step 2: Take Expectations

Now we take the expectation $\mathbb{E}_{\mathcal{D}}$ over this expression. Key observations:

- $f(x)$ and $\mathbb{E}[\widehat{f}]$ are constants w.r.t. $\mathbb{E}_{\mathcal{D}}$.
- $\mathbb{E}_{\mathcal{D}}[\mathbb{E}[\widehat{f}] - \widehat{f}] = \mathbb{E}[\widehat{f}] - \mathbb{E}[\widehat{f}] = 0$.
- Noise ϵ is independent of $\mathcal{D}$, so $\mathbb{E}_{\mathcal{D}}[\epsilon] = 0$ and $\mathbb{E}_{\mathcal{D}}[\epsilon^2] = \sigma_\epsilon^2$.
- Cross terms involving $(\mathbb{E}[\widehat{f}] - \widehat{f})$ will vanish because $\mathbb{E}_{\mathcal{D}}[\mathbb{E}[\widehat{f}] - \widehat{f}] = 0$.

$$\mathbb{E}_{\mathcal{D}}\left[(y - \widehat{f})^2\right] = (f - \mathbb{E}[\widehat{f}])^2 + \mathbb{E}_{\mathcal{D}}\left[(\widehat{f} - \mathbb{E}[\widehat{f}])^2\right] + \sigma_\epsilon^2$$
$$+ 0 + 0 + 0$$

The Final Decomposition

Bias-Variance Decomposition

$$\underbrace{\mathbb{E}_{\mathcal{D}}\left[(y-\widehat{f}(x))^2\right]}_{\text{Expected Prediction Error}} = \underbrace{(f(x)-\mathbb{E}[\widehat{f}(x)])^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[(\widehat{f}(x)-\mathbb{E}[\widehat{f}(x)])^2\right]}_{\text{Variance}} + \underbrace{\sigma_\epsilon^2}_{\text{Irreducible Error}}$$

- **Bias^2:** How wrong is the average prediction of our model? (Accuracy)
- **Variance:** How much do the model's predictions change based on the training data? (Precision)
- **Irreducible Error:** The inherent noise in the data. We cannot reduce this with any model.

Managing the Trade-off in Practice

How to Reduce Bias (fight underfitting):

- Use a more complex model (e.g., higher degree polynomial, deeper tree).
- Add more relevant features.
- Reduce regularization strength.

How to Reduce Variance (fight overfitting):

- Use a simpler model (e.g., linear instead of polynomial).
- Get more training data (the "gold standard" for reducing variance).
- Use feature selection or dimensionality reduction.
- Increase regularization strength (e.g., higher λ in LASSO/Ridge).
- Use ensemble methods (e.g., Bagging, which averages high-variance models).

Summary

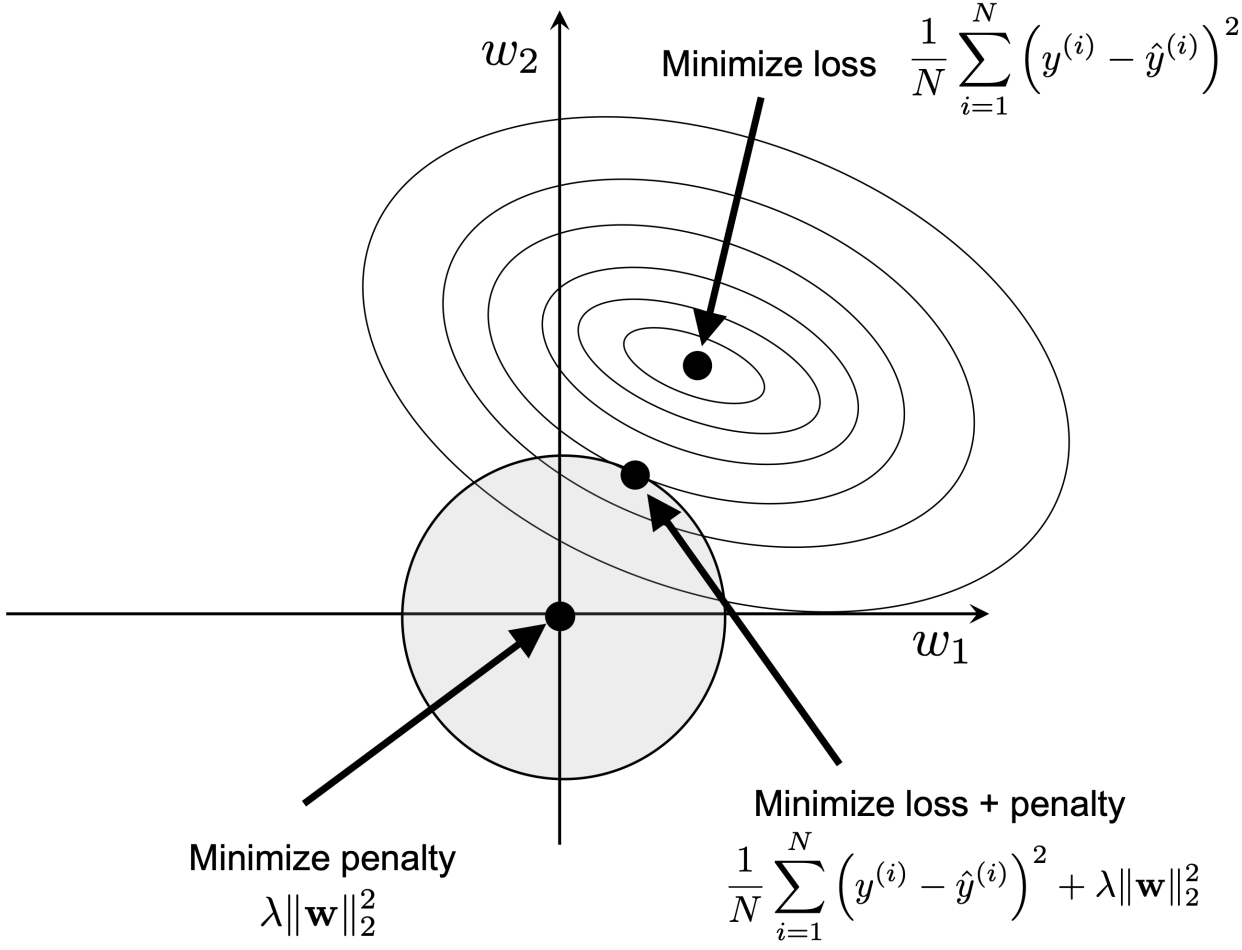
- The Bias-Variance Decomposition provides a powerful framework for understanding generalization error.
- **Total Error = Bias^2 + Variance + Irreducible Error**
- **High Bias** models are too simple and underfit.
- **High Variance** models are too complex and overfit.
- The central challenge in machine learning is to balance this trade-off to minimize total error.
- Practical techniques like regularization, cross-validation, and ensemble methods are direct responses to managing this trade-off.

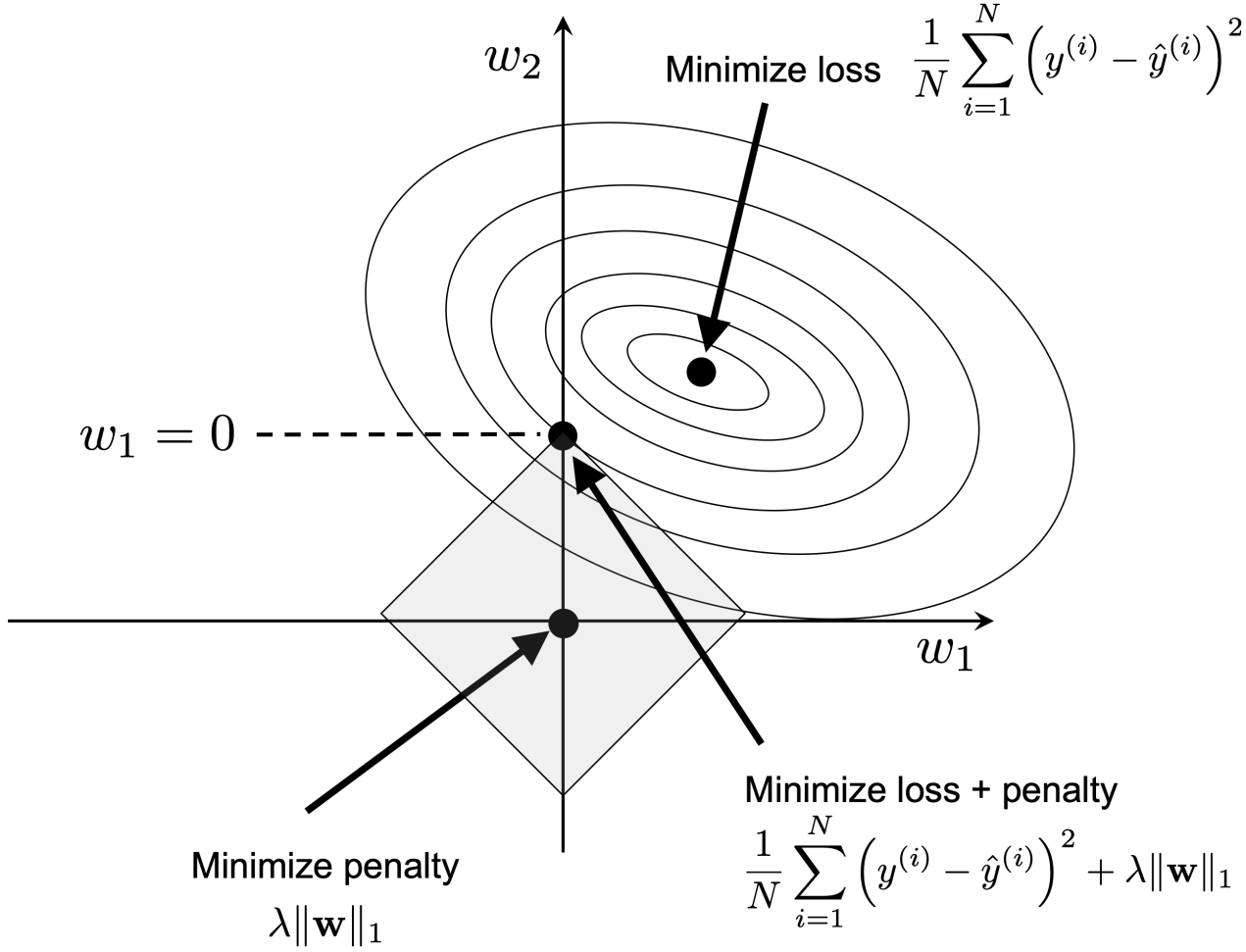
We will discuss regularization, cross-validation, and ensemble methods in detail!

Method 1: Regularization

Change the loss function by adding regularization terms (L2 or L1 regularization)

$$E(w) = E_0(w) + \lambda \sum w_i^2$$





The value of w is decreasing and reduce the complexity effectively. λ is the hyper-parameter and has to be tuned for better performance.

Method 2: Cross-Validation

The Problem with Single Split

- Single train-test split can be **unreliable**
- Performance depends heavily on **which points** end up in test set
- High **variance** in performance estimates
- Might **overfit** to a particular random split

The Cross-Validation Solution

Use all data for both training and testing
through multiple systematic splits

k-Fold Cross-Validation

The k-Fold Process

1. Randomly shuffle dataset and split into **k equal folds**
2. For each fold $i = 1$ to k:
   - Use fold i as **validation set**
   - Use remaining $k - 1$ folds as **training set**
   - Train model and evaluate on validation set
3. Calculate **average performance** across all k iterations

Common Choices of k

- **k=5**: Good balance of computation and reliability
- **k=10**: Very common choice
- **k=n** (Leave-One-Out): Maximum training data but computationally expensive

Key Advantage

Every data point used for validation **exactly once**

Types of Cross-Validation I

Choosing the Right Strategy for Your Data

k-Fold Cross-Validation

- **Most common** method
- Good for most datasets
- Balanced computation and reliability
- *Example: 5-fold or 10-fold*

Leave-One-Out (LOOCV)

- $k = n$ (number of samples)
- Maximum training data
- **Computationally expensive**
- Low bias, high variance

Time Series Cross-Validation

- Respects **temporal order**
- Training on past, testing on future
- Prevents data leakage

Leave-P-Out

- Leave out P samples for testing
- Very computationally intensive
- Exhaustive testing

Repeated k-Fold

Applications: Model Selection and Hyperparameter Tuning I

Hyperparameter Tuning with Cross-Validation

- Test different hyperparameter combinations
- For each combination, run full k-fold CV
- Choose parameters with **best average performance**
- Prevents overfitting to specific validation set

Example: Tuning Regularization Parameter

Applications: Model Selection and Hyperparameter Tuning II

Parameter λ	5-Fold CV Score	Std. Dev.
$\lambda = 0.01$	0.85	0.03
$\lambda = 0.1$	0.88	0.02
$\lambda = 1.0$	**0.92**	**0.01**
$\lambda = 10.0$	0.87	0.04

$\rightarrow$ Choose $\lambda = 1.0$ (highest average score, low variance)

Nested Cross-Validation

- **Outer loop**: Evaluate model performance
- **Inner loop**: Tune hyperparameters
- Provides **unbiased** performance estimate
- Prevents optimistic bias

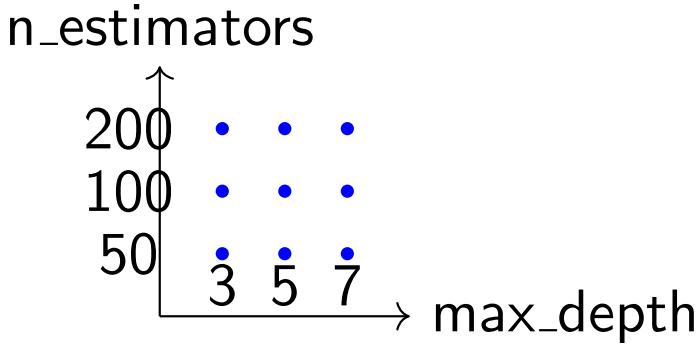
Grid Search: Systematic Hyperparameter Tuning I

Exhaustive Search for Optimal Parameters

What is Grid Search?

- **Systematic approach** to hyperparameter optimization
- Evaluates **all possible combinations** in a predefined grid
- Uses **cross-validation** for robust performance estimation
- Finds the **best parameter set** within the search space

Visualizing the Grid



Tests every intersection point in the grid

Key Features

- **Guaranteed** to find best in g
- **Embarrassingly parallel** - combinations independent
- **Simple** to implement and understand
- Works with any model and scoring metric

Method 3: Ensemble

Combining Multiple Models

The Core Intuition

"None of us is as smart as all of us"

- Individual models make different errors
- When averaged, errors tend to cancel out
- Collective prediction is more stable than any single one

Jellybean Jar Analogy

- 100 people guess jellybeans
- Individual guesses vary widely
- **Average** is remarkably accurate

Key Requirement

Models must make **different types of errors**
(Uncorrelated predictions)

Mathematical Foundation: Variance Reduction I

Why Averaging Works

Ensemble Prediction

For M models with predictions $f_1(x), f_2(x), \ldots, f_M(x)$:

$$f_{\text{avg}}(x) = \frac{1}{M} \sum_{i=1}^{M} f_i(x)$$

Variance of Ensemble

$$\text{Var}[f_{\text{avg}}(x)] = \frac{1}{M^2} \sum_{i=1}^{M} \text{Var}[f_i(x)] + \frac{1}{M^2} \sum_{i \neq j} \text{Cov}[f_i(x), f_j(x)]$$

The Magic of Uncorrelated Models

Mathematical Foundation: Variance Reduction II

Why Averaging Works

If models are uncorrelated (Cov ≈ 0):

$$\text{Var}[f_{\text{avg}}(x)] \approx \frac{1}{M} \times \text{Average Individual Variance}$$

Variance reduces by factor of M!

Bagging: Bootstrap Aggregating I

The Classic Variance Reduction Technique

How Bagging Works

1. Create multiple bootstrap samples
2. Train model on each sample
3. Average predictions (regression)
4. Majority vote (classification)

Bootstrap Sampling

Random sampling with replacement creates diverse training sets

Why Bagging Reduces Variance

- Each model sees different data variation
- Models overfit to different noise patterns
- Averaging cancels out overfitting
- Especially effective for high-variance base models

Ideal for

- Decision trees
- Complex models
- Unstable algorithms

Random Forest: Enhanced Bagging

Adding More Diversity

Beyond Simple Bagging

Random Forest = Bagging + **Random Feature Selection**

- At each split: consider only random subset of features
- Increases diversity among trees
- Further reduces correlation between models

Before Random Forest

Trees often similar, correlated errors

With Random Forest

Trees very different, uncorrelated errors

Double Variance Reduction

1. **Bagging**: Different training data
2. **Random features**: Different split choices

Visual Comparison: Single Model vs Ensemble

Seeing the Variance Reduction

Single Decision Tree

- Overfits to noise
- Unstable boundaries
- High variance

Random Forest (100 Trees)

- Smooth decision boundary
- Stable predictions
- Low variance

Key Observation

Individual trees still overfit, but their **overfitting patterns cancel out** when averaged

Conditions for Effective Variance Reduction I

When Ensembling Works Best

Essential Requirements

- **Diverse base models**: Make different types of errors
- **Uncorrelated predictions**: Errors should cancel out
- **Competent base models**: Individual models should be reasonable
- **Sufficient ensemble size**: More models → more variance reduction

Good Scenario

- 100 different decision trees
- Each trained on different data
- Each using different features
- **Result**: Great variance reduction

Bad Scenario

- 100 identical models
- All trained on same data
- All make same errors
- **Result**: No improvement

Conditions for Effective Variance Reduction II

When Ensembling Works Best

Diversity Creation Strategies

- Different training data (bagging)
- Different features (random subspaces)
- Different algorithms (heterogeneous ensembles)
- Different hyperparameters

Gradient Boosting: Learning from Mistakes I

The Sequential Approach to Boosting

Core Philosophy

"Build models sequentially, each correcting its predecessor's errors"

Analogy: Exam Preparation

1. Take practice test
2. Identify weak areas
3. Study those topics
4. Take another test
5. Repeat until mastery

Key Difference from Bagging

- **Bagging**: Parallel, independent models
- **Boosting**: Sequential, dependent models
- Each model learns from previous mistakes

The Basic Process

Gradient Boosting: Learning from Mistakes II

The Sequential Approach to Boosting

1. Start with simple prediction (mean, median)
2. Calculate errors (residuals)
3. Build model to predict errors
4. Update predictions with small step
5. Repeat many times

Gradient Boosting Algorithm I

Step-by-Step Mathematical Process

Choose a loss function $L(y, a)$, M the number of iterations, and ν the learning rate.

Step 1: Initial Prediction

Start with simple model (e.g., mean for regression):

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

Step 2: For m $= 1$ to M:

Gradient Boosting Algorithm II

Step-by-Step Mathematical Process

1. Compute **pseudo-residuals**:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

2. Fit weak learner $h_m(x)$ (usually small decision tree) to pseudo-residuals (the target)
3. Compute multiplier γ_m:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

4. Update the model:

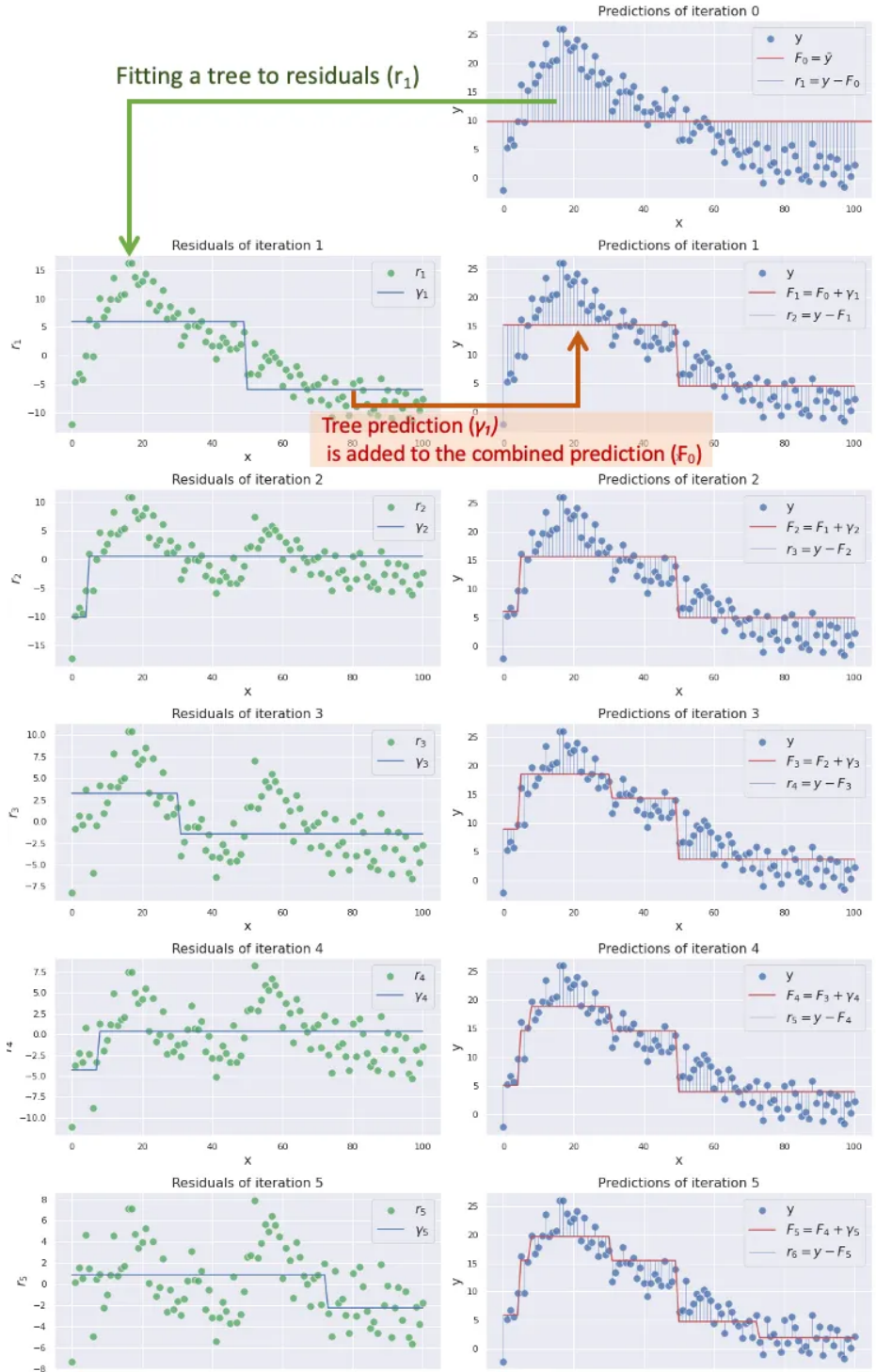
$$F_m(x) = F_{m-1}(x) + \nu\gamma_m h_m(x)$$

Gradient Boosting Algorithm III

Step-by-Step Mathematical Process

Key Parameters

- ν: Learning rate (typically 0.01-0.1)
- M: Number of boosting iterations
- Tree depth: Usually small (3-6)


Predictions of iteration 0
Fitting a tree to residuals (r_1)
Residuals of iteration 1
Predictions of iteration 1
Tree prediction (γ_1)
is added to the combined prediction (F_0)
Residuals of iteration 2
Predictions of iteration 2
Residuals of iteration 3
Predictions of iteration 3
Residuals of iteration 4
Predictions of iteration 4
Residuals of iteration 5
Predictions of iteration 5

Extreme Gradient Boosting (XGBoost) I

Optimized, Scalable, and Winning

What is XGBoost?

- Highly optimized implementation of gradient boosting
- Created by Tianqi Chen in 2016
- Dominates machine learning competitions (Kaggle)
- **10x faster** than traditional gradient boosting

Key Innovations

- **Regularization**: L1 (Lasso) + L2 (Ridge)
- **Parallel processing**: Level-wise tree building
- **Handles missing values** automatically
- **Tree pruning** with max depth
- **Cross-validation** during training

Extreme Gradient Boosting (XGBoost) II

Optimized, Scalable, and Winning

The XGBoost Objective Function

$$\text{Obj}(\theta) = \sum_{i=1}^{n} L(y_i, \widehat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Where $\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 + \alpha \|w\|_1$

- T: Number of leaves, w: Leaf weights
- γ, λ, α: Regularization parameters

XGBoost: Technical Features and Usage

Why It's Faster and Better

Technical Innovations

- **Approximate Algorithm**: For split finding
- **Sparsity-aware**: Handles sparse data
- **Cache-aware**: Optimized memory access
- **Out-of-core**: Handles data larger than memory
- **Weighted Quantile Sketch**: For weighted data

When to Use XGBoost

- **Structured/tabular data**
- **Large datasets**
- **Need for speed** and accuracy
- **Missing values** in data

Section 3: Bayes perspective

Frequentist vs. Bayesian Approach

Frequentist (e.g., AIC, BIC)

- Penalizes log-likelihood by number of parameters.
- Minimizes an information criterion.
- Selects a single "best" model.

Bayesian

- Computes posterior probability of each model.
- Uses marginal likelihood (Bayesian evidence).
- Has a built-in Occam's razor.
- Can average over models (BMA).

The Goal: Posterior Model Probability

In the first stage of Bayes approach, we evaluate the posterior probability for a model. We apply Bayes' Theorem directly to models:

$$p(\mathcal{M}_i|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{M}_i)\, p(\mathcal{M}_i)}{p(\mathcal{D})}$$

- $p(\mathcal{M}_i)$: Prior probability of model $\mathcal{M}_i$.
- $p(\mathcal{D}|\mathcal{M}_i)$: Marginal Likelihood (Model Evidence). The key quantity.
- $p(\mathcal{D})$: Total probability of the data (normalizing constant).
- $p(\mathcal{M}_i|\mathcal{D})$: Posterior probability of model $\mathcal{M}_i$.

Marginal Likelihood: Definition

For a model $\mathcal{M}_i$ with parameters $\boldsymbol{\theta}_i$, the marginal likelihood is the probability of the data averaged over all possible parameter values, weighted by the prior:

$$p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta}_i, \mathcal{M}_i)\, p(\boldsymbol{\theta}_i|\mathcal{M}_i)\, d\boldsymbol{\theta}_i$$

It is the expected value of the likelihood function under the prior.

$$p(\mathcal{D}) = \mathbb{E}_{\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}_i|\mathcal{M}_i)} \left[p(\mathcal{D}|\boldsymbol{\theta}_i, \mathcal{M}_i) \right]$$

The Automatic Occam's Razor

In the second stage of Bayes approach, we choose the models by comparing the evidence.

- **Complex Model** $\mathcal{M}_2$: Prior is spread thinly over a wide range of possible datasets. Its average score (marginal likelihood) for the observed data $\mathcal{D}_0$ is low.
- **Simple Model** $\mathcal{M}_1$: Prior is concentrated on a specific set of datasets. If $\mathcal{D}_0$ falls within this set, its average score is high.
- The marginal likelihood $p(\mathcal{D})$ automatically penalizes unnecessary complexity. $\mathcal{M}_1$ is favored.

Comparing Two Models: Bayes Factor

To compare two models, we look at the posterior odds (let's take the space of models as the random variable):

$$\underbrace{\frac{p(\mathcal{M}_1|\mathcal{D})}{p(\mathcal{M}_2|\mathcal{D})}}_{\text{Posterior Odds}} = \underbrace{\frac{p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{D}|\mathcal{M}_2)}}_{\text{Bayes Factor } BF_{12}} \times \underbrace{\frac{p(\mathcal{M}_1)}{p(\mathcal{M}_2)}}_{\text{Prior Odds}}$$

Bayes Factor

The Bayes Factor is the ratio of the marginal likelihoods of the two models:

$$BF_{12} = \frac{p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{D}|\mathcal{M}_2)}$$

If we assume equal prior odds ($p(\mathcal{M}_1) = p(\mathcal{M}_2)$), the posterior odds equal the Bayes Factor.

Interpreting Bayes Factors

BF_{12}	Evidence in favor of $\mathcal{M}_1$ (against $\mathcal{M}_2$)
1 to 3	Anecdotal / Not worth more than a bare mention
3 to 10	Moderate
10 to 30	Strong
30 to 100	Very Strong
> 100	Decisive / Extreme

Example: $BF_{12} = 15$ means Model 1 is 15 times more probable than Model 2, given the data and equal priors.

The Automatic Occam's Razor: Mathematical Derivation

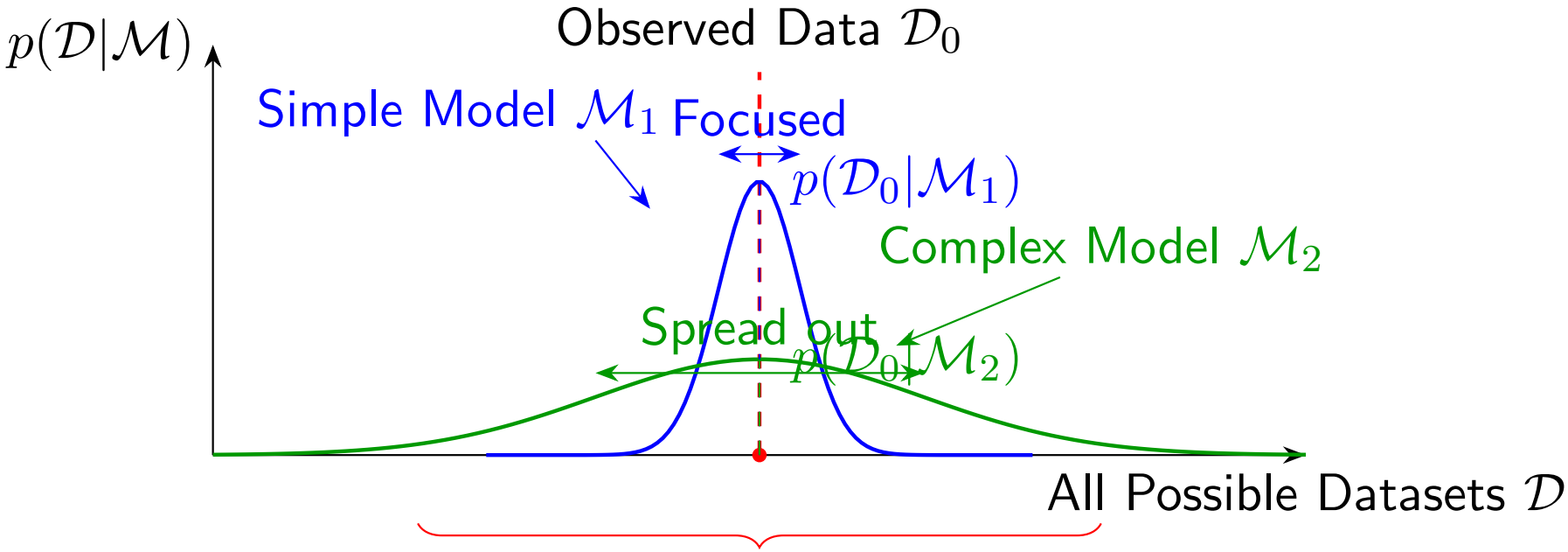
Consider approximating the marginal likelihood for a model $\mathcal{M}$ around the Maximum A Posteriori (MAP) estimate $\widehat{\boldsymbol{\theta}}$.

Laplace Approximation

$$p(\mathcal{D})_{\mathcal{M}} = \int \underbrace{p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M})}_{\text{Likelihood}} \underbrace{p(\boldsymbol{\theta}|\mathcal{M})}_{\text{Prior}} d\boldsymbol{\theta} \approx \underbrace{p(\mathcal{D}|\widehat{\boldsymbol{\theta}}, \mathcal{M})}_{\text{Goodness-of-fit}} \cdot \underbrace{p(\widehat{\boldsymbol{\theta}}|\mathcal{M})(2\pi)^{d/2}|\mathbf{H}|^{-1/2}}_{\text{Occam Factor}}$$

- **Goodness-of-fit**: Best achievable fit (higher is better).
- **Occam Factor**: Penalizes model complexity.
  - $p(\widehat{\boldsymbol{\theta}}|\mathcal{M})$: Prior probability of the best-fit parameters.
  - $|\mathbf{H}|$: Determinant of the Hessian (measures posterior curvature). More parameters $\Rightarrow$ higher d $\Rightarrow$ smaller factor.
  - A more complex model (higher d) will have a smaller Occam Factor, reducing its marginal likelihood unless the fit improves dramatically.

$$\log p(\mathcal{D})_{\mathcal{M}} \approx \log p(\mathcal{D}|\widehat{\boldsymbol{\theta}}, \mathcal{M}) + \log p(\widehat{\boldsymbol{\theta}}|\mathcal{M}) + \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{H}|$$



Complex model spreads probability over wider range of datasets

Why Select Just One Model?

Selecting a single "best" model ignores model uncertainty. This can lead to overconfident predictions and inferior generalization. The Bayesian solution is to average over all models.

Bayesian Model Averaging (BMA)

For a new data point x^*, we want to predict y^*. Instead of using one model, we use all of them, weighted by their posterior probability:

$$p(y^*|x^*, \mathcal{D}) = \sum_{i=1}^{K} p(y^*|x^*, \mathcal{M}_i, \mathcal{D})\, p(\mathcal{M}_i|\mathcal{D})$$

- This is a mixture distribution.
- The prediction is an average of the predictions from all models.
- The weight for each model's prediction is its posterior probability $p(\mathcal{M}_i|\mathcal{D})$.
- BMA accounts for model uncertainty and typically provides better calibrated and more robust predictions than any single model.

The Big Challenge: Calculating the Integral

$$p(\mathcal{D}|\mathcal{M}_i) = \int p(\mathcal{D}|\boldsymbol{\theta}_i, \mathcal{M}_i)\, p(\boldsymbol{\theta}_i|\mathcal{M}_i)\, d\boldsymbol{\theta}_i$$

This integral is high-dimensional and analytically intractable for most models of interest. **Solution: Approximations**

- **Laplace Approximation:** Approximate the posterior as a Gaussian.
- **Variational Inference (VI):** Turn the integral into an optimization problem.
- **MCMC Methods:** Draw samples from the posterior. The harmonic mean of likelihoods is a poor estimator. Better methods include:
  - Chib's method
  - Bridge sampling
  - Nested sampling

Summary

- Bayesian model selection is based on posterior model probabilities.
- The key quantity is the marginal likelihood $p(\mathcal{D}|\mathcal{M}_i)$, which integrates over parameter uncertainty.
- It features an automatic Occam's razor, naturally penalizing complex models.
- Models are compared using Bayes Factors (BF_{12}).
- Instead of selecting one model, we can perform Bayesian Model Averaging (BMA) for more robust predictions.
- The main practical challenge is computing the marginal likelihood, which requires sophisticated approximations.