Lecture 8: Model Assessment & Selection, Generalization Theory

Readings: ESL (Ch. 7, 9.1), ISL (Ch. 5-6), Bach (Ch. 4); code

Soon Hoe Lim

September 18, 2025

Outline

Overview

⚠ In our last lectures, we have unlocked incredible power of kernel methods but this immense power comes with a hidden danger (why?).

❓ How to understand and build models that generalize well to unseen data?

I: Practical Model Assessment & Selection (ESL Ch. 7, ISL Ch. 5)
Goal: Learn *how* to assess and select models in practice.

- Understanding prediction error and the bias-variance tradeoff
- Cross-validation and resampling methods
- Information criteria (AIC, BIC) for model selection
- Bootstrap methods and uncertainty quantification

II: Theory for Obtaining Generalization Bounds (Bach Ch. 4 & Appendix)
Goal: Understand *why* certain models generalize better through learning theory.

- Convex surrogates for intractable problems
- Risk decomposition and the sources of error
- Rademacher complexity for generalization bounds

The Fundamental Challenge: Test vs Training Error

Definition 1: Training vs Test Error

Given training set[a] $\mathcal{T} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ and learned model $\hat{f}$:
- **Training Error:** $\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$.
- **Test Error (Generalization Error):** $\text{Err}_{\mathcal{T}} = \mathbb{E}_{X^0, Y^0}[L(Y^0, \hat{f}(X^0)) \mid \mathcal{T}]$ where (X^0, Y^0) is a new test point independent of $\mathcal{T}$. This conditional test error refers to the error for the specific training set $\mathcal{T}$. Taking another average over *all* randomness gives the *expected test error*: $\text{Err} = \mathbb{E}[\text{Err}_{\mathcal{T}}]$.

[a] We have used the notation D_N to denote the dataset in earlier lectures. Here we are using $\mathcal{T}$ to emphasize that it is a training set. This also matches the notation of ESL.

- **Regression:** $L(Y, \hat{f}(X)) = (Y - \hat{f}(X))^2$ (squared error)
- **Classification:** $L(G, \hat{G}(x)) = \mathbb{I}[G \neq \hat{G}(x)]$ (0-1 loss), or, $L(G, \hat{G}(x)) = -2\sum_{k=1}^{K} \mathbb{I}(G = k) \log \hat{p}_k(X) = -2 \log \hat{p}_G(X) = 2 \times \text{NLL}$

💡 Estimation of $\text{Err}_{\mathcal{T}}$ is the goal, but Err is more amenable to analysis. Let's focus on the regression setting for ease of exposition.

Bias-Variance Tradeoff

The Core Challenge: We want to minimize test error, but we can only observe training error. Low training error $\neq$ low test error due to **overfitting**.

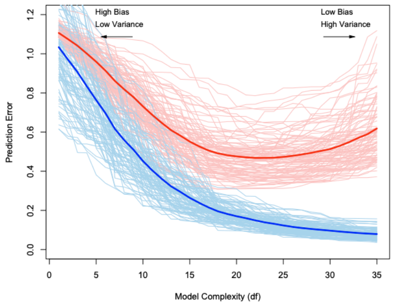


FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\text{err}}$*, while the light red curves show the conditional test error* $\text{Err}_{\mathcal{T}}$ *for* 100 *training sets of size* 50 *each, as the model complexity is increased. The solid curves show the expected test error* Err *and the expected training error* $\text{E}[\overline{\text{err}}]$.

Model Assessment vs. Model Selection

Definition 2: Model Assessment vs Model Selection

- **Model Selection:** Choose the best model from a set of candidates by comparing their estimated test performance
- **Model Assessment:** Estimate the test error (generalization error) of our final chosen model to understand its performance on new data

⚠ **Golden Rule:** The same data should not be used for both model selection and final assessment! This leads to overly optimistic performance estimates.

❓ How to estimate the expected test error for a model?

In Practice

If we are in a data-rich situation, then we can randomly divide[1] the dataset into three sets (a training set, a validation set, a test set).

In practice:

- We have **multiple candidate models** (LDA, logistic regression, SVM, etc.) (fit on a training set)
- Models have **tuning parameters** (regularization strength λ, kernel bandwidth, etc.) (use a validation set)
- We need **reliable estimates** of how well our final model will perform (on a test set)

❓ But what if we do not have sufficient[2] data to do the split? How should we estimate the expected test error for a given model f?

[1] Difficult to give a general rule on how to choose the number of samples in each of the three sets, as this depends on the SNR in the data and the training sample size.

[2] Again, it is difficult to give a general rule on how much training data is enough, as this could depend on the SNR of the underlying function and the model complexity.

The Optimism of Training Error

- **Training error:** $\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$, i.e. loss on the same training responses (x_i, y_i). Typically *too optimistic* (see Exercise 8.2).
- **In-sample error:**

$$\text{Err}_{in} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{Y_i^0}[L(Y_i^0, \hat{f}(x_i)) \mid \mathcal{T}].$$

  Average loss on new responses Y_i^0 at the same training inputs x_i.
- **Test (extra-sample) error:**

$$\text{Err}_{\mathcal{T}} = \mathbb{E}_{X^0, Y^0}[L(Y^0, \hat{f}(X^0)) \mid \mathcal{T}].$$

  Average loss on fully new input-output pairs (X^0, Y^0).

The optimism in $\overline{\text{err}}$ is easier to understand if we focus on the *in-sample error*.

Definition 3: Optimism

The **optimism** is the gap between in-sample and training error: $\text{op} = \text{Err}_{in} - \overline{\text{err}}$, and the average optimism is $\omega = \mathbb{E}_{\mathbf{y}}[\text{op}]$.

An Important Relation

Fundamental relationship:

$$\mathbb{E}_{\mathbf{y}}[\mathrm{Err}_{in}] = \mathbb{E}_{\mathbf{y}}[\overline{\mathrm{err}}] + \omega.$$

General form. For squared error loss[a], one can show that (see Exercise 8.1):

$$\omega = \frac{2}{N} \sum_{i=1}^{N} \mathrm{Cov}(\hat{y}_i, y_i).$$

[a] In fact, also for 0-1 and other loss functions.

Special case. Consider the addtive error model $Y = f(X) + \epsilon$, where the noise ϵ is independent of X, $\mathbb{E}[\epsilon] = 0, \mathrm{Var}(\epsilon) = \sigma^2 I$. For the linear models with d parameters fitted by least squares, $\omega = \frac{2\sigma^2 d}{N}$ (the overfitting amount you derived for an exercise in Lecture 2).

💡 Each parameter "uses up" $2\sigma^2/N$ worth of optimism. More complex models are more optimistic.

Information Criteria: AIC, BIC, and C_p

Prediction error can be estimated by correcting the **training error** with an estimate of the **optimism**, i.e. the gap between Err_{in} and $\overline{\text{err}}$.

Definition 4: General Form

$$\widehat{\text{Err}}_{in} = \overline{\text{err}} + \widehat{\omega}$$

where $\widehat{\omega}$ estimates the average optimism $\omega = \mathbb{E}[\text{Err}_{in} - \overline{\text{err}}]$.

Specific Criteria

- **Mallows' C_p (linear regression):** $C_p = \overline{\text{err}} + \frac{2d}{N}\hat{\sigma}^2$, with d = number of parameters and $\hat{\sigma}^2$ = estimated noise variance.
- **Akaike Information Criterion (AIC):** $\text{AIC} = -2\log\mathcal{L}(\hat{\theta}) + 2d$, where $\mathcal{L}$ is the maximized likelihood. For Gaussian models[3]: $\text{AIC} = N\log(2\pi\hat{\sigma}^2) + \frac{\text{RSS}}{\hat{\sigma}^2} + 2d$.
- **Bayesian Information Criterion (BIC):** $\text{BIC} = -2\log\mathcal{L}(\hat{\theta}) + d\log N$. Since $\log N > 2$ for $N > 7$, BIC applies a stronger penalty than AIC.

[3] For model comparison in Gaussian models, the constant terms cancel, giving $\text{AIC} \propto \frac{\text{RSS}}{\hat{\sigma}^2} + 2d$. Scaling by $\frac{\hat{\sigma}^2}{N}$ yields $C_p = \overline{\text{err}} + \frac{2d\hat{\sigma}^2}{N}$, showing the explicit $\frac{2d\hat{\sigma}^2}{N}$ factor.

Information Criteria: Remarks and Caveats

- **AIC:** Asymptotically equivalent to leave-one-out cross-validation under certain conditions
  - Tends to select models with complexity close to optimal for prediction
  - Can overfit in small samples
- **BIC:** Asymptotically consistent, arises in the Bayesian approach to model selection (see ESL Ch. 7.7)
  - If true model is in candidate set, BIC selects it with probability $\to 1$
  - More conservative (selects simpler models) than AIC
  - Better for interpretation, worse for prediction when truth is complex

Practical Guidelines

- **Use AIC** when primary goal is prediction accuracy
- **Use BIC** when seeking to identify "true" parsimonious model
- Both require likelihood-based models and proper parameter counting
- Not directly applicable to non-likelihood methods (SVM, trees, etc.)

⚠ **Limitation:** Information criteria rely on asymptotic approximations and may be unreliable with small samples or misspecified models.

Cross-Validation: The Gold Standard

Cross-validation (CV) directly estimates test error Err by simulating the train/test process:

- Split data into training and validation sets
- Train on training set, evaluate on validation set
- Repeat with different splits to get stable estimate

K-Fold Cross-Validation

```
1: Input: Dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, number of folds $K$
2: Randomly partition $\mathcal{D}$ into $K$ disjoint folds: $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_K$
3: for $k = 1, 2, \ldots, K$ do
4:     $\mathcal{D}_{\text{train}}^{(k)} \leftarrow \mathcal{D} \setminus \mathcal{D}_k$        ▷ Training = all except fold $k$
5:     $\mathcal{D}_{\text{val}}^{(k)} \leftarrow \mathcal{D}_k$        ▷ Validation = fold $k$
6:     Train model: $\hat{f}^{(-k)} \leftarrow \text{Learn}(\mathcal{D}_{\text{train}}^{(k)})$
7:     Compute validation error: $\text{Err}_k \leftarrow \frac{1}{|\mathcal{D}_k|} \sum_{(x,y)\in\mathcal{D}_k} L(y, \hat{f}^{(-k)}(x))$
8: end for
9: Return: $\text{CV}_K(\hat{f}) \leftarrow \frac{1}{K} \sum_{k=1}^{K} \text{Err}_k$
```

It turns out that CV estimates effectively the average error Err (not the conditional error $\text{Err}_{\mathcal{T}}$); see Ch 7.12 in ESL.

Cross-Validation: Variants and Bias-Variance Properties

- **5-fold or 10-fold CV:** Good bias-variance tradeoff, computationally feasible. 10-fold CV most common.
- **Leave-One-Out CV (LOOCV)[4]:** Use $K = N$, nearly unbiased but high variance.
- **Stratified CV:** Maintains class proportions in each fold (for classification).
- **Generalized CV (GCV):** Approximation to LOOCV for linear fitting under squared-error loss (see Exercise 8.4 to see how GCV can be related to AIC).

Bias:

- CV uses $(N - N/K)$ training samples vs. N for final model
- Small K (few folds) $\rightarrow$ more bias (pessimistic estimates)
- Large K (many folds) $\rightarrow$ less bias

Variance:

- Small K $\rightarrow$ less variance (fewer, more different estimates to average)
- Large K $\rightarrow$ more variance (many, highly correlated estimates)
- LOOCV often has very high variance

[4]See Exercise 8.3 to derive LOOCV for linear smoothers and kernel ridge regression.

Cross-Validation for Model Selection

Model Selection via Cross-Validation

```
1: **Input:** Dataset $\mathcal{D}$, candidate models $\mathcal{M}_1, \ldots, \mathcal{M}_m$
2: for each model $\mathcal{M}_j$ do
3:     Compute $\text{CV}_K(\mathcal{M}_j)$ using $K$-fold cross-validation
4: end for
5: $\hat{j} \leftarrow \arg\min_j \text{CV}_K(\mathcal{M}_j)$        ▷ Select best model
6: **Return:** Best model $\mathcal{M}_{\hat{j}}$
```

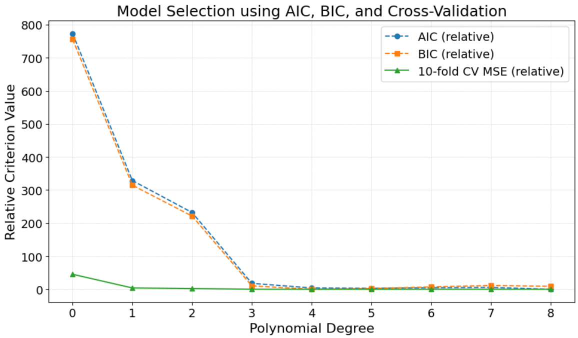
When both selecting and assessing models:

1. **Outer loop:** Split data into train/test
2. **Inner loop:** Use CV on training set to select best model
3. **Assessment:** Evaluate selected model on held-out test set

This prevents **selection bias** - overly optimistic estimates from using the same data for both selection and assessment.

⚠ **Common Mistake:** Using the CV score from model selection as the final performance estimate. This is biased! Need separate test set or nested CV.

Example: Selecting A Polynomial Regression Model



Degree	AIC	BIC	CV_MSE
0	1335.76	1342.36	46.30
1	891.15	901.04	4.97
2	794.39	807.58	3.15
3	580.27	596.76	1.06
4	566.52	**586.31**	1.00
5	565.40	588.49	1.00
6	567.21	593.60	1.00
7	568.14	597.82	1.00
8	**562.42**	595.40	**0.96**

Bootstrap Methods

💡 If we can't get new samples from the population, create "new" samples by resampling from our data.

Bootstrap Procedure

```
1: Input: Original dataset $\mathcal{T} = \{(x_1, y_1), \dots, (x_N, y_N)\}$, $z_i := (x_i, y_i)$
2: for $b = 1, 2, \dots, B$ do
3:     Create $\mathcal{T}^{*b}$ by sampling $N$ points from $\mathcal{T}$ with replacement
4:     Train model: $\hat{f}^{*b} \leftarrow \text{Learn}(\mathcal{T}^{*b})$
5:     Compute statistic of interest on $\hat{f}^{*b}$
6: end for
7: Analyze distribution of statistics across bootstrap samples
```

The bootstrap is another resampling method for estimating risk. It involves drawing B "bootstrap samples" of size N from the training data *with replacement*.

💡 Each bootstrap sample omits, on average[5], 36.8% of the original data points. We can use these "out-of-bag" (OOB) points to form an error estimate.

[5] Why 36.8%? See Exercise 8.5.

Bootstrap Procedure

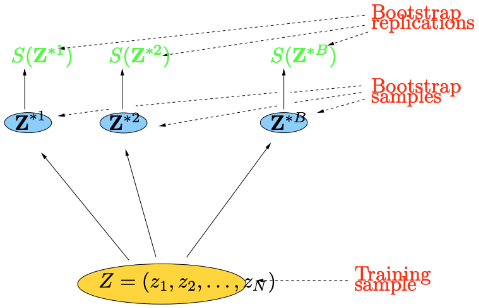


FIGURE 7.12. *Schematic of the bootstrap process. We wish to assess the statistical accuracy of a quantity* $S(\mathbf{Z})$ *computed from our dataset.* B *training sets* $\mathbf{Z}^{*b}$, $b = 1, \ldots, B$ *each of size* N *are drawn with replacement from the original dataset. The quantity of interest* $S(\mathbf{Z})$ *is computed from each bootstrap training set, and the values* $S(\mathbf{Z}^{*1}), \ldots, S(\mathbf{Z}^{*B})$ *are used to assess the statistical accuracy of* $S(\mathbf{Z})$.

Bootstrap Error Estimation

The **leave-one-out bootstrap (LOOB)** error estimate is:

$$\widehat{\text{Err}}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}}\ell(y_i, \hat{f}^{*b}(x_i))$$

where C^{-i} is the set of indices of the bootstrap samples b that do not contain observation i.

The LOOB estimate can be biased upwards because models are trained on smaller, less diverse datasets. The .632 estimators correct for this.

Definition 5: The .632 Estimator

A weighted average of the (optimistic) empirical risk and the (pessimistic) LOOB error estimate:

$$\widehat{\text{Err}}^{(.632)} = 0.368 \cdot \overline{\text{err}} + 0.632 \cdot \widehat{\text{Err}}^{(1)}$$

This can fail in heavily overfit situations where $\overline{\text{err}} \approx 0$.

The .632+ Estimator: An Improved Version

Definition 6: The .632+ Estimator

$$\widehat{\mathrm{Err}}^{(.632+)} = (1-\hat{w})\cdot\overline{\mathrm{err}} + \hat{w}\cdot\widehat{\mathrm{Err}}^{(1)}.$$

The weight $\hat{w}$ adapts to the relative overfitting rate.

The adaptive weight is:

$$\hat{w} = \frac{0.632}{1-0.368\cdot\hat{R}}$$

where the **relative overfitting rate** is:

$$\hat{R} = \frac{\widehat{\mathrm{Err}}^{(1)} - \overline{\mathrm{err}}}{\hat{\gamma} - \overline{\mathrm{err}}}$$

and $\hat{\gamma}$ is the **no-information error rate**: $\hat{\gamma} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} L(y_i, \hat{f}(x_j))$ (estimates the error rate when responses are randomly paired with predictors).

- When $\hat{R} = 0$ (no overfitting), $\hat{w} = 0.632$, reducing to .632 estimator.
- When $\hat{R} \to 1$ (severe overfitting), $\hat{w} \to 1$, trusting only bootstrap estimate.

Bootstrap and Maximum Likelihood (See ESL Ch. 8)

Maximum Likelihood Estimation (MLE)

- MLE: $\hat{\theta} = \arg\max_\theta \prod_{i=1}^{N} f(y_i;\theta)$
- Under regularity: $\hat{\theta}$ is approximately normal with variance $\propto 1/N$
- Variance can be estimated using *Fisher information:* $I(\theta) = \mathbb{E}\left[-\frac{\partial^2}{\partial\theta^2} \log f(Y;\theta)\right]$

Bootstrap Inference

- Resample data with replacement, refit model $\Rightarrow$ distribution of $\hat{\theta}^*$
- Provides standard errors, confidence intervals, bias estimates
- Practical when analytic formulas (like Fisher info) are hard

💡 MLE gives asymptotic theory, bootstrap gives practical inference.

How to Estimate Test Error: Method Comparison[7]

Method	**Pros**	**Cons**
Validation Set	Simple to understand and implement, fast	Reduces effective sample size, can be unreliable with small datasets
AIC/BIC/C_p	Fast computation, well-established theory, good for linear models	Requires likelihood or OLS, assumes model is approximately correct, can be unreliable in small samples
10-Fold CV	Widely applicable, minimal assumptions, direct test error estimate	Computationally intensive ($10\times$ cost), can be unstable with small datasets
LOOCV[6]	Nearly unbiased for test error, deterministic result	Very high variance, extremely expensive, can be unstable
Bootstrap	More stable than LOOCV, provides uncertainty estimates, handles complex models well	More complex to implement, can still have bias issues, requires many bootstrap samples

[6] LOOCV resembles the jackknife: both leave one point out. The difference is purpose: LOOCV estimates prediction error, jackknife estimates bias/variance. Historically, jackknife came before bootstrap.

[7] However, see Exercise 8.6 for a paradox.

Practical Guidelines

1. **For most applications:** Use 10-fold cross-validation[8]
   - Excellent bias-variance tradeoff
   - Works with any learning algorithm
   - Widely accepted and understood
2. **For linear models with likelihood:** Consider AIC/BIC as alternatives
   - AIC for prediction-focused applications
   - BIC for model interpretation and parsimony
3. **For very small datasets ($N < 100$):** Use LOOCV or bootstrap
   - Every sample counts
   - Higher variance acceptable given limited data
4. **For model selection + assessment:** Use nested CV or train/validation/test split
   - Prevents optimistic bias
   - Essential for honest performance[9] reporting

[8]Despite the widespread use of CV and its seeming simplicity, its operating properties remain opaque; see, e.g., `https://arxiv.org/abs/2104.00673`.

[9]To assess generalization, we need both reliable estimation methods and suitable performance metrics. Different metrics capture different task requirements. Common metrics include accuracy, precision/recall, F1 score, and ROC–AUC for classification; MSE, MAE, and R^2 for regression.

Applying Our Toolkit to Generalized Additive Models

- Linear models can describe additive, linear relationships between features but have limited ability to capture interactions automatically.
- Fully non-parametric approaches suffers from the curse of dimensionality, and require exponentially more data as dimension increases.

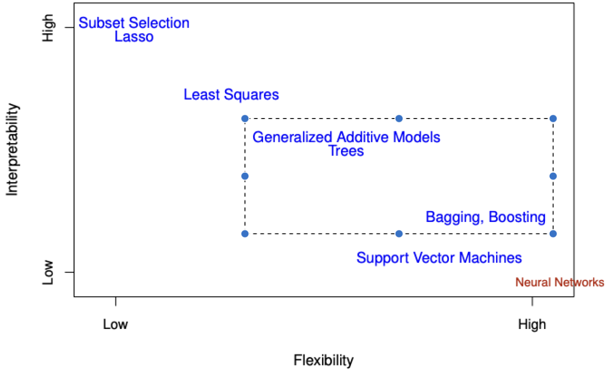
Real-world relationships are often:

- Non-linear: $f(x) = x^2 + \sin(x)$
- Interactive: $f(x_1, x_2) = x_1 \cdot x_2$
- Piecewise: Different behavior in different regions

❓ Can we build flexible models that adapt their complexity to the data in a *natural, interpretable way*?

💡 Let's look at a class of models that sit between the simple linear models and the 'black box' kernel methods: **Generalized Additive Models (GAMs)**.

Interpretability vs. Prediction



As models become flexible, interpretability drops.
Today, we will focus on GAMs (which strike a sweet spot in the chart above), and leave trees, bagging and boosting to next lecture.

Occam Razor principle: Everything has to be kept as simple as possible, but not simpler (Albert Einstein).

Additive Models: A Flexible Compromise

Definition 7: Additive Model

In the regression setting, an additive model has the form:

$$\mathbb{E}[Y|X_1, \dots, X_p] = \alpha + \sum_{j=1}^{p} f_j(X_j)$$

where each f_j is an unknown smooth univariate function.

Key advantages:

- **Interpretability:** Effect of X_j on Y captured by $f_j(X_j)$ alone
- **Flexibility:** Each f_j can be any smooth (nonparametric) function, fit using a scatterplot smoother (e.g., cubic smoothing spline or kernel smoother)
- **Dimensionality:** Avoids curse by modeling univariate functions
- **Identifiability:** Require $\sum_{i=1}^{N} f_j(x_{ij}) = 0$ for each j

Extensions: Can include selected interactions: $f_{jk}(X_j, X_k)$

How GAMs are Fit: The Backfitting Algorithm

The goal is to minimize RSS[10]: $\sum_{i=1}^{N}\left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij})\right)^2$

Backfitting Algorithm

```
1: Initialize: $\hat{\alpha} = \bar{y} = \frac{1}{N}\sum_{j=1}^{N} y_j$, $\hat{f}_j \equiv 0$ for $j = 1, \ldots, p$
2: repeat
3:     for $j = 1, \ldots, p$ do
4:         Compute partial residuals: $r_{ij} = y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})$
5:         Smooth partial residuals: $\hat{f}_j \leftarrow \mathcal{S}_j[\{(x_{ij}, r_{ij})\}_{i=1}^{N}]$
6:         Center: $\hat{f}_j(x) \leftarrow \hat{f}_j(x) - \frac{1}{N}\sum_{i=1}^{N} \hat{f}_j(x_{ij})$
7:     end for
8: until convergence
```

Common smoothers $\mathcal{S}_j$: cubic smoothing splines, local regression, kernel methods.

The smoothers have their own hyperparameters, which we must tune using CV.

[10]For classification, we instead fit via maximum penalized likelihood, often using the backfitting algorithm on a modified response variable (iteratively reweighted least squares).

Beyond Gaussian: The Exponential Family

Definition 8: Exponential Family

A random variable Y follows an exponential family distribution if its density can be written as:

$$f(y;\theta,\phi) = \exp\left\{\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right\}$$

where θ is the **natural parameter**, ϕ is the **dispersion parameter**, and $b(\cdot)$, $a(\cdot)$, $c(\cdot)$ are known functions.

Key Properties:

- $\mathbb{E}[Y] = \mu = b'(\theta)$ (mean function)
- $\text{Var}(Y) = a(\phi)\cdot b''(\theta) = a(\phi)\cdot V(\mu)$ where $V(\mu)$ is the **variance function**

Common Examples:

- **Gaussian:** $\theta = \mu$, $b(\theta) = \theta^2/2$, $V(\mu) = 1$
- **Poisson:** $\theta = \log(\mu)$, $b(\theta) = e^\theta$, $V(\mu) = \mu$
- **Binomial:** $\theta = \log(\mu/(1-\mu))$, $b(\theta) = \log(1+e^\theta)$, $V(\mu) = \mu(1-\mu)$

Generalized Linear Models (GLMs)

Definition 9: Generalized Linear Model

A GLM consists of three components:

1. **Random Component:** $Y \sim$ Exponential Family with mean μ
2. **Systematic Component:** Linear predictor $\eta = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$
3. **Link Function:** $g(\mu) = \eta$ where g is a monotonic, differentiable function

Canonical Link: When $g(\mu) = \theta$ (natural parameter), we have:

- **Gaussian:** Identity link $g(\mu) = \mu$
- **Poisson:** Log link $g(\mu) = \log(\mu)$
- **Binomial:** Logit link $g(\mu) = \log(\mu/(1-\mu))$

Key Insight: GLMs extend linear models by allowing non-Gaussian responses and non-linear relationships through the link function, while maintaining linear structure in parameters.

Binary Classification: Logistic Regression

Setting: Response $Y \in \{0, 1\}$ follows Binomial$(1, p)$ distribution

Definition 10: Logistic Regression Model

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

where $p(x) = \Pr(Y = 1|X = x)$ and the left side is the **log-odds** or **logit**.

Equivalent forms:

- **Probability:** $p(x) = \frac{e^{\beta_0+\sum_{j=1}^{p}\beta_j x_j}}{1+e^{\beta_0+\sum_{j=1}^{p}\beta_j x_j}} = \frac{1}{1+e^{-(\beta_0+\sum_{j=1}^{p}\beta_j x_j)}}$
- **Linear predictor:** $\eta = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$
- **Logistic function:** $p(x) = \frac{1}{1+e^{-\eta}}$

Properties:

- $p(x) \in (0, 1)$ always
- S-shaped curve: gradual near boundaries, steep in middle
- Linear decision boundary: $\eta = 0 \Leftrightarrow p(x) = 0.5$

From GLMs to Generalized Additive Models

Limitation of GLMs: Still assume linear relationships in the linear predictor

Definition 11: Generalized Additive Model (GAM)

A GAM replaces the linear predictor in a GLM with an additive predictor:

$$g(\mu) = \alpha + \sum_{j=1}^{p} f_j(X_j)$$

where g is the link function and each f_j is a smooth univariate function.

For Binary Classification (Logistic GAM): $\log\left(\frac{p(x)}{1-p(x)}\right) = \alpha + \sum_{j=1}^{p} f_j(x_j)$

For Poisson Regression (Poisson GAM): $\log(\mu) = \alpha + \sum_{j=1}^{p} f_j(x_j)$

💡 GAMs combine the flexibility of nonparametric smoothing with the distributional flexibility of GLMs, while maintaining additivity for interpretability.

⚠ For any GAM, the main task stays the same: using tools like CV to select the right flexibility for each f_j, and using the bootstrap to understand uncertainty.

Generalized Additive Models: The Complete Framework

Definition 12: Generalized Additive Model (GAM)

A GAM extends GLM by keeping the same three-component structure:

1. **Random Component:** $Y \sim$ Exponential Family (unchanged from GLM)
2. **Systematic Component:** Additive predictor $\eta = \alpha + \sum_{j=1}^{p} f_j(X_j)$
3. **Link Function:** $g(\mu) = \eta$ (same link functions as GLM)

- **Gaussian GAM:** $\mathbb{E}[Y|X] = \alpha + \sum_{j=1}^{p} f_j(X_j)$ (identity link)
- **Logistic GAM:** $\log\left(\frac{p(x)}{1-p(x)}\right) = \alpha + \sum_{j=1}^{p} f_j(x_j)$ (logit link)
- **Poisson GAM:** $\log(\mu) = \alpha + \sum_{j=1}^{p} f_j(x_j)$ (log link)

What stays the same: Distributional assumptions, link function theory, likelihood-based inference

What changes: Linear predictor $\rightarrow$ Additive predictor with smooth functions

GAMs = GLM flexibility for distributions + nonparametric flexibility for relationships

Advantages and Limitations of GAMs

Advantages:

- **Interpretability:** Each f_j can be plotted and interpreted separately
- **Flexibility:** Automatic adaptation to non-linear patterns
- **Inference:** Pointwise confidence intervals for each f_j
- **Generality:** Works with any exponential family distribution

Limitations:

- **Additivity assumption:** No interactions unless explicitly included
- **Curse persists:** Including interactions $f_{jk}(X_j, X_k)$ brings back dimensionality issues
- **Model selection:** Choosing which variables to smooth vs. keep linear
- **Extrapolation:** Smoothers can behave poorly outside training range

When to use GAMs: When you need interpretable models with moderate flexibility, have sufficient data for smoothing, and additivity assumption is reasonable. Model assessment and selection are crucial!

Exercises

Exercise 8

1. **Average Optimism in Training Error.** Solve Exercise 7.4 and 7.5 in ESL.
2. **Training vs. Test Error in Linear Regression.** Solve Exercise 2.9 in ESL.
3. **LOOCV for Linear Smoothers and kernel ridge regression (KRR).**

   (a) Consider the KRR with kernel matrix K and regularization parameter $\lambda > 0$. Show that the fitted values can be written as a linear smoother $\hat{\mathbf{y}} = S\mathbf{y}$ where $S = K(K + \lambda I)^{-1}$, $K_{ij} = k(x_i, x_j)$ and $\mathbf{y} = (y_1, \dots, y_N)$.

   (b) Using the linear smoother representation, derive the leave-one-out cross-validation (LOOCV) formula: $y_i - \hat{y}_i^{-i} = \frac{y_i - \hat{y}_i}{1 - S_{ii}}$.

   (c) Solve Exercise 7.3 in ESL to derive LOOCV for general linear smoothers.
4. C_p**/AIC vs. GCV.** Solve Exercise 7.7 in ESL. Discuss how the GCV formula can be applied to the KRR in Exercise 3 using the smoother matrix.
5. **How Likely Is an Observation to Appear in a Bootstrap Sample?** Solve Exercise 2 in Ch. 5.4 of ISL.

Exercises

Exercise 8

6. **The Majority Classifier Paradox.** Suppose that you are given a dataset of 100 samples, consisting of 50 positives and 50 negatives. Consider a learning algorithm that always predicts each new sample as the *majority class in the training set* (breaking ties by random guessing if the training set is perfectly balanced). Estimate the expected error rate of this model under the following evaluation schemes:

   (a) 10-fold cross-validation
   (b) Leave-one-out cross-validation (LOOCV)
   (c) Hold-out with a 70%/30% split
   (d) Hold-out with a 50%/50% split
   (e) The out-of-bag (OOB), .632, and .632+ bootstrap estimates.

   Compare the results and briefly explain the message of this problem.

7. **[Experimental] Best Subset Analysis.** Solve Exercise 7.9 in ESL.

Appendix: From Practice to Theory

- **Part I (Practice):** Data-driven heuristics for estimating test error
  - Criteria like AIC/BIC, resampling methods like CV and bootstrap
- **Limitation:** These methods usually work well in practice, but do not explain *why* models generalize.
- **Part II (Theory):** Statistical Learning Theory
  - Why does empirical risk approximate expected risk (if at all)?
  - How does model complexity affect generalization?

We will only attempt to give a brief introduction to statistical learning theory[11] following Ch. 4 in Bach here. See, e.g., `https://cs.nyu.edu/~mohri/mlbook/` or Shalev-Shwartz & Ben-David (2014) for more results and the omitted details.

[11] This is an optional topic, useful for exposure, but it will not be tested in the exam.

The Learning Problem: Formal Setup

Definition 13: Statistical Learning Framework

- **Data:** $(x_1, y_1), \ldots, (x_N, y_N)$ drawn i.i.d. from unknown distribution P on $\mathcal{X} \times \mathcal{Y}$
- **Goal:** Learn function $f : \mathcal{X} \to \mathcal{Y}$ that minimizes expected risk:

$$R(f) = \mathbb{E}_{(x,y)\sim P}[\ell(y, f(x))]$$

- **Bayes Optimal:** $f^* = \arg\min_f R(f)$ with $R^* = R(f^*)$
- **Our Focus:** Methods based on **Empirical Risk Minimization (ERM)**: $\hat{f} = \arg\min_{f \in \mathcal{F}} \hat{R}(f) = \arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i))$

We have focused on the regression setting so far. But how about classification?

Key Challenge: For binary classification with 0-1 loss, this is a combinatorial optimization problem. We need better approaches!

The Convexification Strategy

For binary classification ($\mathcal{Y} = \{-1, 1\}$) with 0-1 loss:

1. Learn real-valued function $g : \mathcal{X} \to \mathbb{R}$ using convex surrogate loss $\Phi(yg(x))$
2. Make predictions via $f(x) = \text{sign}(g(x))$

Benefits:

- **Computational:** Convex optimization (global optimum, efficient algorithms)
- **Theoretical:** Enables Rademacher complexity analysis
- **Practical:** Allows gradient-based methods and principled regularization

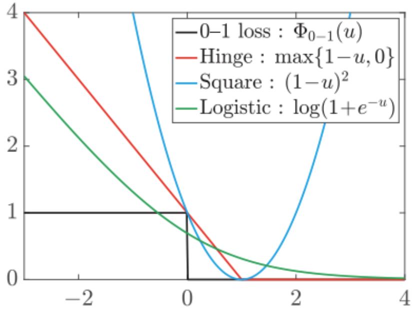
For binary classification, consider surrogate losses of the form $\Phi(yg(x))$ where $yg(x)$ is the **margin**.

- **Margin > 0:** Correct classification with confidence
- **Margin < 0:** Misclassification

Convexification with Calibrated Surrogate Losses

A surrogate loss Φ is **calibrated** if minimizing $\mathbb{E}[\Phi(yg(x))]$ leads to the Bayes optimal classifier. All losses below can be shown to be calibrated:

- **Hinge Loss (SVM):** $\Phi(u) = \max(0, 1-u)$
- **Logistic Loss:** $\Phi(u) = \log(1+e^{-u})$
- **Exponential Loss (AdaBoost):** $\Phi(u) = e^{-u}$
- **Squared Loss:** $\Phi(u) = (1-u)^2$



Risk Decomposition: Sources of Error

We consider loss functions that are defined for real-valued outputs (for binary classification problems we will use a surrogate loss).

For any ERM estimator $\hat{f} \in \mathcal{F}$ trained on N samples:

$$R(\hat{f}) - R^* = \underbrace{R(\hat{f}) - \inf_{f \in \mathcal{F}} R(f)}_{\text{Estimation Error}} + \underbrace{\inf_{f \in \mathcal{F}} R(f) - R^*}_{\text{Approximation Error}}$$

- **Approximation Error:** Fundamental limitation of the model class $\mathcal{F}$
  - How well can the *best possible* function in $\mathcal{F}$ approximate f^*?
  - Only reduced by choosing more flexible $\mathcal{F}$ (more complex models)
  - Independent of sample size N
- **Estimation Error:** Error from finite sample learning
  - How much worse is our empirical solution $\hat{f}$ vs. the best in class?
  - Decreases with more data (typically $O(1/\sqrt{N})$ or better)
  - Increases with model complexity (richer $\mathcal{F}$)

The estimation error depends on how well empirical risk approximates true risk across the *entire function class*.

Setting and Goal

- We consider N i.i.d. random variables $z_1, \ldots, z_N \in \mathcal{Z}$, and a class $\mathcal{H}$ of (measurable) functions $h : \mathcal{Z} \to \mathbb{R}$.
- In the learning setting, each $z = (x, y)$ is an input–label pair. The function class is defined by the loss functions induced by predictors $f \in \mathcal{F}$, i.e.:

$$\mathcal{H} = \left\{ (x, y) \mapsto \ell(y, f(x)), f \in \mathcal{F} \right\}.$$

- Our goal is to use some complexity measure of class $\mathcal{H}$ to bound the **generalization gap**, i.e.

$$\sup_{f \in \mathcal{F}} \{R(f) - \hat{R}(f)\},$$

  where $R(f) = \mathbb{E}[\ell(y, f(x))]$, $\hat{R}(f) = \frac{1}{N} \sum_{i=1}^N \ell(y_i, f(x_i))$.
- Equivalently, in terms of $\mathcal{H}$:

$$\sup_{h \in \mathcal{H}} \left(\mathbb{E}[h(z)] - \frac{1}{N} \sum_{i=1}^N h(z_i) \right),$$

  where $\mathbb{E}[h(z)]$ is expectation with respect to a fresh $z \sim \mathrm{Law}(z_i)$.
- In particular, the estimation error can be controlled by the generalization gap, since $R(\hat{f}) - \inf_{f \in \mathcal{F}} R(f) \leq 2 \sup_{f \in \mathcal{F}} \left\{ R(f) - \hat{R}(f) \right\}$.

Rademacher Complexity: Definition

💡 Functions that can memorize random noise are prone to overfitting real data. Can we measure this 'overfitting potential'? See blackboard for a motivation.

Definition 14: Rademacher Complexity

Consider N i.i.d. samples $z_1, \ldots, z_N \in \mathcal{Z}$ and a class $\mathcal{H}$ of functions from $\mathcal{Z}$ to $\mathbb{R}$. The Rademacher complexity of $\mathcal{H}$ is:

$$\mathcal{R}_N(\mathcal{H}) = \mathbb{E}_{z,\varepsilon}\left[\sup_{h \in \mathcal{H}} \frac{1}{N}\sum_{i=1}^{N} \varepsilon_i h(z_i)\right]$$

where $\varepsilon_i \overset{\text{iid}}{\sim}$ Rademacher(± 1) are independent of the data.

- **Random Noise Test:** How well can functions in $\mathcal{H}$ fit random noise (ε_i)?
- **Complexity Measure:** Rich function classes can fit noise well $\Rightarrow$ high $\mathcal{R}_N(\mathcal{H})$
- **Sample Size Effect:** More data makes it harder to fit noise $\Rightarrow \mathcal{R}_N(\mathcal{H})$ typically decreases as N increases
- **Dimension Independent:** Often gives dimension-free bounds

Symmetrization: The Key Lemma

Connection to Generalization: Functions that can fit random noise well are prone to overfitting real data. Rademacher complexity is data-dependent and quantifies this overfitting potential.

To connect the estimation error to Rademacher complexity, we need a key technical tool.

Lemma 1: Symmetrization Lemma

For any function class $\mathcal{H}$, $\mathbb{E}\left[\sup_{h\in\mathcal{H}}\left(\frac{1}{N}\sum_{i=1}^{N} h(z_i) - \mathbb{E}[h(z)]\right)\right] \leq 2\mathcal{R}_N(\mathcal{H})$ and $\mathbb{E}\left[\sup_{h\in\mathcal{H}}\left(\mathbb{E}[h(z)] - \frac{1}{N}\sum_{i=1}^{N} h(z_i)\right)\right] \leq 2\mathcal{R}_N(\mathcal{H})$.

Proof.
See blackboard. □

Rademacher Bound

Generalization gap depends on model complexity and sample size.

Theorem 1: Rademacher Bound

Let $\mathcal{F}$ be a function class and ℓ be a loss function with $|\ell(y, f(x)| \leq \ell_\infty$. Define the loss function class $\mathcal{H} = \{(x, y) \mapsto \ell(y, f(x)) : f \in \mathcal{F}\}$. Then for any $\delta > 0$, with probability at least $1-\delta$, $\sup_{f\in\mathcal{F}}(R(f)-\hat{R}(f)) \leq 2\mathcal{R}_N(\mathcal{H})+\ell_\infty\sqrt{\frac{\log(1/\delta)}{2N}}$.

Proof.

See blackboard. □

This bounds the worst-case generalization gap over the entire function class $\mathcal{F}$. Since our ERM solution $\hat{f} \in \mathcal{F}$, this bound applies to $\hat{f}$ as well.

The following result will be useful later.

Proposition 1: Contraction Principle

If $\ell(y, \cdot)$ is L-Lipschitz and $\ell(y, 0) = 0$ for all y, then for loss class $\mathcal{H} = \{\ell(\cdot, f(\cdot)) : f \in \mathcal{F}\}$, $\mathcal{R}_N(\mathcal{H}) \leq L \cdot \mathcal{R}_N(\mathcal{F})$.

Rademacher Complexity: Key Properties

Rademacher complexity can often be bounded using **norms**.

Proposition 2: Linear Predictions (Ball-Constrained)

For $\mathcal{F} = \{f_\theta(x) = \theta^\top \phi(x) : \|\theta\|_2 \leq D\}$: $\mathcal{R}_N(\mathcal{F}) = \frac{D}{N}\mathbb{E}\left[\left\|\sum_{i=1}^N \varepsilon_i \phi(x_i)\right\|_2\right]$

If $\mathbb{E}[\|\phi(X)\|_2^2] \leq R^2$, then: $\mathcal{R}_N(\mathcal{F}) \leq \frac{DR}{\sqrt{N}}$.

Proof.

See blackboard. □

💡 **Dimension-Free Bound:** The bound $DR/\sqrt{N}$ does not depend on the dimension of $\phi(x)$! Applicable to infinite-dimensional spaces (e.g., RKHS).

This is a norm-based bound — it depends on the norm of weights and data, not parameter dimension/count.

Norm-Based Generalization Bound

Theorem 2: Generalization Bound for Linear Predictors

Consider:

- Linear predictors: $\mathcal{F} = \{f_\theta(x) = \theta^\top \phi(x) : \|\theta\|_2 \le D\}$
- Loss function is L-Lipschitz and bounded (by ℓ_∞)
- Bounded features: $\mathbb{E}[\|\phi(X)\|_2^2] \le R^2$
- ERM estimator $\hat{f} = \arg\min_{f \in \mathcal{F}} \hat{R}(f)$

Then $\mathbb{E}[R(\hat{f})] - \inf_{f \in \mathcal{F}} R(f) \le \frac{4LDR}{\sqrt{N}}$. Moreover, for any $\delta > 0$, with probability $\ge 1 - \delta$, $R(\hat{f}) - \inf_{f \in \mathcal{F}} R(f) \le \frac{4LDR}{\sqrt{N}} + \ell_\infty \sqrt{\frac{\log(1/\delta)}{2N}}$.

Proof.

See blackboard. □

- Bound scales with $L \cdot D \cdot R$ (loss smoothness $\times$ model complexity $\times$ data scale)
- **Dimension-free:** Works for infinite-dimensional $\phi(x)$ (kernels!)
- Others (not covered here): VC dimension bound, PAC-Bayes bound, etc.

Key Takeaways

- **Practice:** Cross-validation, AIC/BIC, bootstrap methods give usable estimates of test error.
- **Theory:** Statistical learning theory explains why generalization is possible. The generalization bound[12] of Theorem 2 explains why models generalize even in infinite dimensions or with far more parameters than samples. It provides intuition about why *controlling capacity (regularization, small weights, etc.) helps generalization*.
- Both perspectives are complementary:
  - Empirical tools guide model assessment and selection.
  - Theoretical tools justify and bound their performance.

💡 **Controlling generalization error = unifying theme.**

[12] In practice (esp. for deep learning), the bound is often loose and is not the best tool to numerically predict generalization error. It's more valuable for theoretical insight than practical error estimation.