

Lecture 6: Generalization Theory, Model Assessment & Selection

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

Soon Hoe Lim

September 11, 2025

Outline

- ① Part I: Theoretical Foundations
- ② Part II: Practical Model Assessment & Selection
- ③ Summary
- ④ Exercises

Overview

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

Part I: Theoretical Foundations (Bach Ch. 4)

Goal: Understand *why* certain models generalize better through statistical learning theory.

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

Part II: Practical Methods (ESL Ch. 7, ISL Ch. 5)

Goal: Learn *how* to assess and select models in practice.

- ▶ Bias-variance tradeoff and model complexity
- ▶ Cross-validation and resampling methods
- ▶ Information criteria (AIC, BIC) for model selection

The Learning Problem: Formal Setup

Definition 1: Statistical Learning Framework

- ▶ **Data:** $(x_1, y_1), \dots, (x_n, y_n)$ drawn i.i.d. from unknown distribution P on $\mathcal{X} \times \mathcal{Y}$
- ▶ **Goal:** Learn function $f : \mathcal{X} \rightarrow \mathcal{Y}$ 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)**:

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \hat{R}(f) = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(x_i))$$

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 $g : \mathcal{X} \rightarrow \mathbb{R}$ 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

Classical Surrogate Losses

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

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

A surrogate loss Φ is **calibrated** if minimizing $\mathbb{E}[\Phi(yg(x))]$ leads to the Bayes optimal classifier. All losses below are 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**: $\Phi(u) = (1 - u)^2$

Sources of Error in Learning

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:

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

- ▶ **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})

Uniform Deviation and Concentration

Estimation error is controlled by uniform deviation over the function class:

$$\mathbb{E}[R(\hat{f})] - \inf_{f \in \mathcal{F}} R(f) \leq 2\mathbb{E} \left[\sup_{f \in \mathcal{F}} |R(f) - \hat{R}(f)| \right]$$

Proposition 1: McDiarmid's Inequality

Let $Z = \sup_{f \in \mathcal{F}} (R(f) - \hat{R}(f))$ where loss is bounded: $\ell(y, f(x)) \in [0, \ell_\infty]$. Then,

$$P(Z \geq \mathbb{E}[Z] + t) \leq \exp \left(-\frac{2nt^2}{\ell_\infty^2} \right),$$

and with probability $\geq 1 - \delta$:

$$\sup_{f \in \mathcal{F}} (R(f) - \hat{R}(f)) \leq \mathbb{E} \left[\sup_{f \in \mathcal{F}} (R(f) - \hat{R}(f)) \right] + \ell_\infty \sqrt{\frac{\log(1/\delta)}{2n}}.$$

The Problem: We still need to bound $\mathbb{E}[\sup_{f \in \mathcal{F}} |R(f) - \hat{R}(f)|]$. This is where Rademacher complexity comes in!

Rademacher Complexity: Definition

Definition 2: Rademacher Complexity

Given function class \mathcal{H} and i.i.d. sample z_1, \dots, z_n :

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

where $\varepsilon_i \stackrel{\text{iid}}{\sim} \text{Rademacher}(\pm 1)$ 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:** Unlike VC dimension, 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 quantifies this overfitting potential.

Lemma 1: Symmetrization Lemma

For any function class \mathcal{H} :

$$\mathbb{E} \left[\sup_{h \in \mathcal{H}} \left(\frac{1}{n} \sum_{i=1}^n h(z_i) - \mathbb{E}[h(Z)] \right) \right] \leq 2\mathcal{R}_n(\mathcal{H}).$$

Proof Sketch

1. **Ghost Sample:** Introduce independent copy z'_1, \dots, z'_n with same distribution
2. **Expectation Trick:** $\mathbb{E}[h(Z)] = \mathbb{E}[\frac{1}{n} \sum_{i=1}^n h(z'_i)]$ for any h
3. **Symmetry:** Since z_i, z'_i have same distribution:

$$\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n (h(z_i) - h(z'_i))$$

has same distribution as $\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \varepsilon_i (h(z_i) - h(z'_i))$

4. **Contraction:** Final step uses properties of Rademacher random variables

Rademacher Complexity: Key Properties

Proposition 2: Contraction Principle


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

Proposition 3: Linear Functions

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: $\mathcal{R}_n(\mathcal{F}) \leq \frac{DR}{\sqrt{n}}$.

 **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).

Main Generalization Theorem

Theorem 1: Generalization Bound for Linear Predictors

Consider:

- ▶ Linear predictors: $\mathcal{F} = \{f_\theta(x) = \theta^\top \phi(x) : \|\theta\|_2 \leq D\}$
- ▶ L -Lipschitz loss function
- ▶ Bounded features: $\mathbb{E}[\|\phi(X)\|_2^2] \leq R^2$
- ▶ ERM estimator $\hat{f} = \arg \min_{f \in \mathcal{F}} \hat{R}(f)$

Then $\mathbb{E}[R(\hat{f})] - \inf_{f \in \mathcal{F}} R(f) \leq \frac{4LDR}{\sqrt{n}}$.

With probability $\geq 1 - \delta$: $R(\hat{f}) - \inf_{f \in \mathcal{F}} R(f) \leq \frac{4LDR}{\sqrt{n}} + \ell_\infty \sqrt{\frac{\log(1/\delta)}{2n}}$.

- ▶ Rate $O(1/\sqrt{n})$ is **minimax optimal** for this setting
- ▶ 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!)

Regularization and Fast Rates

From Hard Constraints to Soft Penalties

In practice, we often use regularized ERM instead of hard constraints:

$$\hat{\theta}_\lambda = \arg \min_{\theta} \left[\frac{1}{n} \sum_{i=1}^n \ell(y_i, \theta^\top \phi(x_i)) + \frac{\lambda}{2} \|\theta\|_2^2 \right]$$

Proposition 4: Fast Rates for Strongly Convex Losses

Assume:

- ▶ Loss $\ell(y, \cdot)$ is convex and L -Lipschitz
- ▶ Features bounded: $\|\phi(x)\|_2 \leq R$ almost surely

Then $\mathbb{E}[R(\hat{\theta}_\lambda)] \leq \inf_{\theta} \left\{ R(\theta) + \frac{\lambda}{2} \|\theta\|_2^2 \right\} + \frac{24L^2R^2}{\lambda n}$.

Optimization of λ

- ▶ **Estimation Error:** $O(\frac{1}{\lambda n})$ - decreases with larger λ
- ▶ **Regularization Bias:** Increases with λ (penalizes complex solutions)
- ▶ **Optimal Choice:** $\lambda^* \propto \frac{1}{\sqrt{n}}$ gives overall rate $O(\frac{1}{\sqrt{n}})$


From Theory to Practice

Linear models can achieve good theoretical guarantees. But in practice:

- ▶ We have **multiple candidate models** (LDA, logistic regression, SVM, etc.)
- ▶ Models have **tuning parameters** (regularization strength λ , kernel bandwidth, etc.)
- ▶ We need **reliable estimates** of how well our final model will perform

Definition 3: 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 of our final chosen model to understand its real-world performance

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

The Fundamental Goal: Minimizing Test Error

Definition 4: Training vs Test Error

Given training set $\mathcal{T} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ and learned model \hat{f} :

Training Error: $\text{err}_{\mathcal{T}} = \frac{1}{N} \sum_{i=1}^N L(y_i, \hat{f}(x_i))$

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} .

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

- ▶ **Regression:** $L(y, \hat{f}(x)) = (y - \hat{f}(x))^2$ (squared error)
- ▶ **Classification:** $L(y, \hat{f}(x)) = \mathbb{I}[y \neq \hat{f}(x)]$ (0-1 loss)

The Optimism of Training Error

Definition 5: Optimism

The **optimism** is the expected difference between test and training error: $\text{op} = \mathbb{E}_{\mathcal{T}}[\text{Err}_{\mathcal{T}}] - \mathbb{E}_{\mathcal{T}}[\text{err}_{\mathcal{T}}]$.

This gives us the fundamental relationship: $\mathbb{E}_{\mathcal{T}}[\text{Err}_{\mathcal{T}}] = \mathbb{E}_{\mathcal{T}}[\text{err}_{\mathcal{T}}] + \text{op}$

Optimism for Linear Models. For a linear model with d parameters fitted by least squares: $\text{op} = \frac{2d}{N} \sigma^2$.

💡 Each parameter "uses up" $2\sigma^2/N$ worth of degrees of freedom.

More complex models (larger d) have higher optimism:

- ▶ **Linear model:** $\text{op} \propto d$
- ▶ **Neural networks:** $\text{op} \propto \# \text{ parameters}$
- ▶ **Nonparametric methods:** $\text{op} \propto \text{effective degrees of freedom}$

Information Criteria: AIC, BIC, and C_p

These methods estimate test error by correcting the training error with a complexity penalty.

Definition 6: General Form

$$\widehat{\text{Err}} = \text{Training Error} + \text{Penalty}(\text{Model Complexity})$$

Specific Criteria

Mallows' C_p (for OLS regression): $C_p = \text{err}_{\mathcal{T}} + \frac{2d}{N} \hat{\sigma}^2$ where $\hat{\sigma}^2$ is noise variance estimate (typically from full model).

Akaike Information Criterion (AIC): $\text{AIC} = -2 \log \mathcal{L}(\hat{\theta}) + 2d$ For Gaussian errors, $-2 \log \mathcal{L} \propto \text{RSS}$, so $\text{AIC} \approx C_p$.

Bayesian Information Criterion (BIC): $\text{BIC} = -2 \log \mathcal{L}(\hat{\theta}) + d \log N$ Since $\log N \geq 2$ for $N \geq 8$, BIC penalizes complexity more than AIC.


Information Criteria: Properties and Usage

Theoretical Properties

- ▶ **AIC:** Asymptotically equivalent to cross-validation
 - ▶ Tends to select models with complexity close to optimal for prediction
 - ▶ Can overfit in small samples
- ▶ **BIC:** Consistent model selection
 - ▶ If true model is in candidate set, BIC selects it with probability $\rightarrow 1$
 - ▶ 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 directly estimates test error 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 $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$, number of folds K
- 2: Randomly partition \mathcal{D} into K disjoint folds: $\mathcal{D} = \mathcal{D}_1 \cup \dots \cup \mathcal{D}_K$
- 3: **for** $k = 1, 2, \dots, K$ **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:** $\text{CV}_K \leftarrow \frac{1}{K} \sum_{k=1}^K \text{Err}_k$

Cross-Validation: Variants and Properties

- ▶ **5-fold or 10-fold CV:** Good bias-variance tradeoff, computationally feasible. 10-fold CV most common.
- ▶ **Leave-One-Out CV (LOOCV):** $K = n$, nearly unbiased but high variance
- ▶ **Stratified CV:** Maintains class proportions in each fold (for classification)
- ▶ **Time series CV:** Respects temporal order (no future information leak)

Bias-Variance of CV Estimates

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

Cross-Validation for Model Selection


Model Selection via Cross-Validation

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

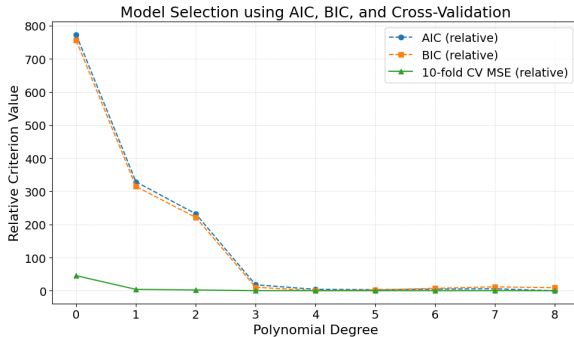
When both selecting and assessing models:

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

The Bootstrap

Bootstrap Principle

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

Bootstrap Procedure

- 1: **Input:** Original dataset $\mathcal{T} = \{(x_1, y_1), \dots, (x_n, y_n)\}$
- 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

- ▶ Each bootstrap sample contains $\approx 63.2\%$ of original data points
- ▶ Remaining $\approx 36.8\%$ are "out-of-bag" (OOB) samples
- ▶ Bootstrap mimics sampling variability from population

Bootstrap for Prediction Error Estimation

Naive Bootstrap (Biased)

Train \hat{f}^{*b} on bootstrap sample \mathcal{T}^{*b} , test on original data \mathcal{T} :

$\widehat{\text{Err}}_{\text{boot}} = \frac{1}{B} \frac{1}{n} \sum_{b=1}^B \sum_{i=1}^n L(y_i, \hat{f}^{*b}(x_i))$ **Problem:** Training and test sets overlap
→ overly optimistic

Leave-One-Out Bootstrap

For each point (x_i, y_i) , average predictions from bootstrap samples that don't contain it: $\widehat{\text{Err}}_{\text{LOO-boot}} = \frac{1}{n} \sum_{i=1}^n \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} L(y_i, \hat{f}^{*b}(x_i))$ where $C^{-i} = \{b : (x_i, y_i) \notin \mathcal{T}^{*b}\}$.

Proposition 5: Bootstrap Estimator

Combines training error and LOO-bootstrap error: $\widehat{\text{Err}}^{(.632)} = 0.368 \cdot \text{err}_{\mathcal{T}} + 0.632 \cdot \widehat{\text{Err}}_{\text{LOO-boot}}$

The weights come from the expected overlap: $\mathbb{E}[|\mathcal{T}^{*b} \cap \mathcal{T}|/n] \approx 0.632$.

Theoretical vs Practical: Bridging the Gap?

Theoretical Insights:

- ▶ **Convex surrogates** make hard problems tractable
- ▶ **Rademacher complexity** gives dimension-free generalization bounds
- ▶ **Regularization** can improve rates from $O(1/\sqrt{n})$ to $O(1/n)$
- ▶ **Linear models** have strong theoretical guarantees

Practical Tools:

- ▶ **Cross-validation** works broadly without strong assumptions
- ▶ **Information criteria** efficient for likelihood-based models
- ▶ **Bootstrap** provides flexible uncertainty quantification
- ▶ **Nested CV** prevents selection bias in assessment

Method Comparison and Recommendations

Method	Pros	Cons
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	Nearly unbiased for test error, deterministic result	Very high variance, extremely expensive ($n\times$ cost), 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
Validation Set	Simple to understand and implement, fast	Reduces effective sample size, can be unreliable with small datasets

Practical Guidelines

1. **For most applications:** Use 10-fold cross-validation
 - ▶ Excellent bias-variance tradeoff
 - ▶ Works with any learning algorithm
 - ▶ Widely accepted and understood
2. **For linear models with likelihood:** Consider AIC/BIC as faster 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 reporting

Exercise 1

1. **Calibration.** Show that the logistic loss and the exponential loss are calibrated. Construct an example of a non-calibrated surrogate loss.
2. **Rademacher Complexity.** Solve Exercise 4.8 in Bach.
3. **Massart's Lemma.** Solve Exercise 4.9 in Bach.
4. **Generalization Bound for Kernel Methods.** Extend Theorem 1 to kernel methods.
5. **Best Subset Analysis.** Solve Exercise 7.9 in ESL.