Machine Learning and Optimization

Lecture 2: Modeling Decisions— Hypothesis Spaces, Losses, and Regularization

CSCI 4140/6960 — Spring 2026

Course website: `cs.rpi.edu/~gittea/.../mlandopt.html`

Lecture 2 Roadmap

Challenges of Optimization in ML

What is Machine Learning?

Example: Binary Classification

Support Vector Machines

Generalizing to Supervised Learning

Optimization Challenges in Machine Learning

Constrained vs Unconstrained:

$$\underbrace{\min_{\boldsymbol{\theta} \in \Theta} J(\boldsymbol{\theta})}_{\text{constrained}} \quad \text{vs} \quad \underbrace{\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})}_{\text{unconstrained}}$$

Key challenges:

- Dealing with constraints (e.g., NNLS vs OLS from Lecture 1)
- Matching the objective to the task (which norm? which loss?)
- Problems are lack closed-form solutions and are often difficult
- Scale: Large-scale linear regression with millions of parameters

Today's focus: How do we formulate ML problems as optimization?

What is Machine Learning?

"Finding predictable patterns from data"

More precisely: Finding a function $f(x) = y$ that is

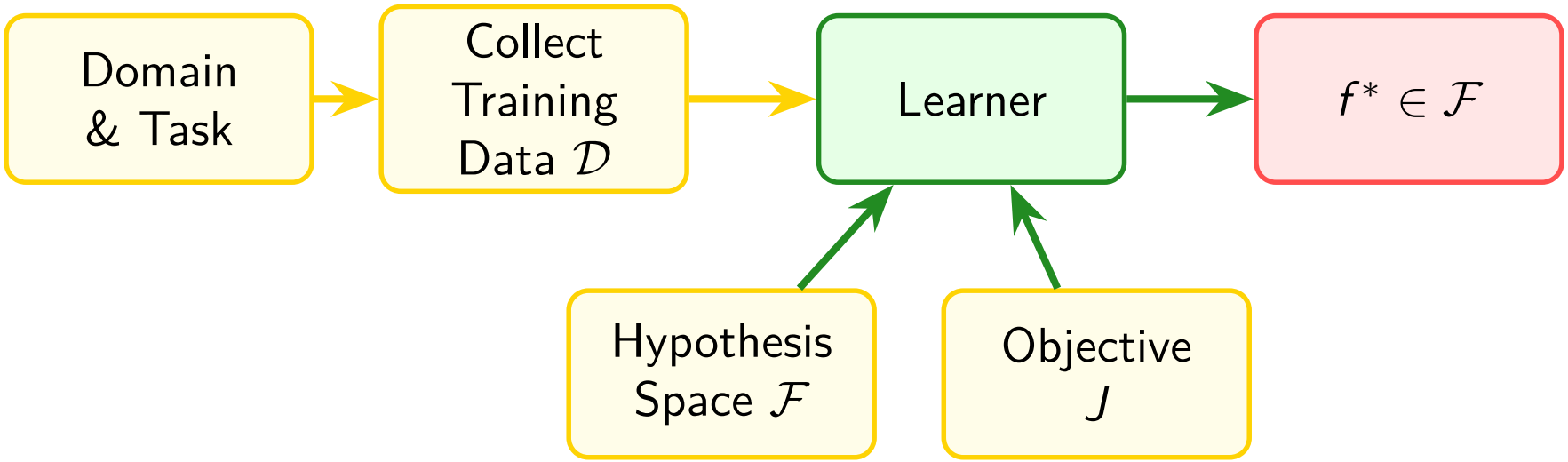
- **accurate** in the presence of randomness
- learned from **randomly selected observations** (x_i, y_i)

Assumptions that make ML possible:

1. There *is* an accurate function to find
2. That function *can be learned* from data

Question for next lecture: What does "accurate in the presence of randomness" mean mathematically?

The Machine Learning Pipeline



Today: We'll make this pipeline concrete by instantiating it for a binary classification task, and showing how we develop the SVM objective step-by-step.

Where Optimization Enters ML

We saw last time that the learner solves:

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

Today we ask: What does this look like for a specific task?

Challenges in ML optimization:

- Choosing $\mathcal{F}$ and the loss ℓ to match the task
- Problems can be large-scale, high-dimensional
- Exact solutions often unavailable

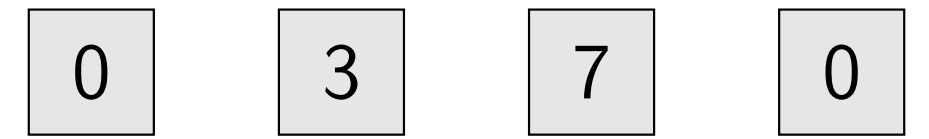
Our plan: Work through a concrete example (binary classification) to see how modeling choices become optimization problems.

Example: Digit Recognition (MNIST)

Task: Predict which digit is in a grayscale image

Data: $\mathbf{x} \in \mathbb{R}^{28 \times 28} \sim \mathbb{R}^d$ where $d = 28 \times 28 = 784$
Grayscale image of digits 0-9

Simplification: Predict if digit is 0 (class -1) or not (class $+1$)



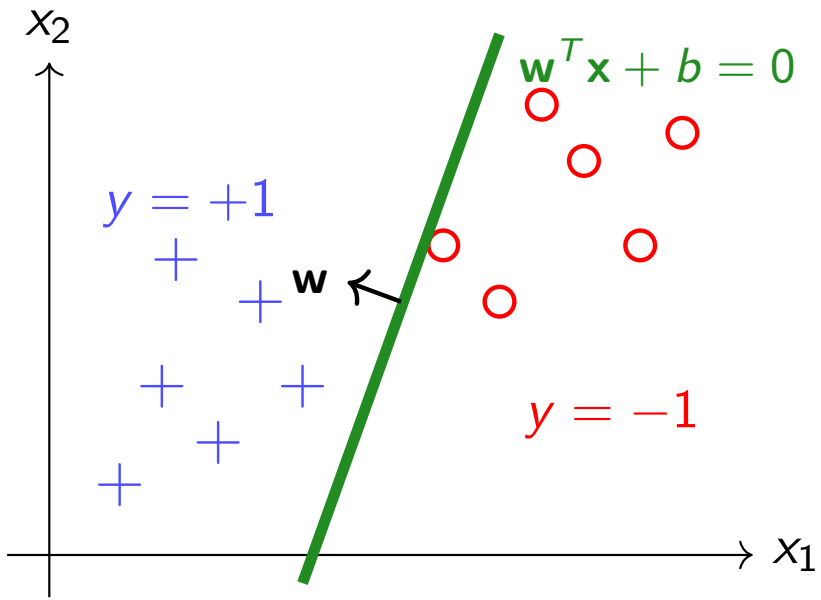
$y = -1$ $y = +1$ $y = +1$ $y = -1$

Goal: Given training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, find function
$f : \mathbb{R}^d \to \{-1, +1\}$

To learn a function, we have to make assumptions about f

Hypothesis: Linear Separability

Assumption: The two classes can be separated by a hyperplane



Hypothesis space:

$$\mathcal{F} = \left\{ f_{\mathbf{w},b} : \mathbb{R}^d \to \{-1, +1\} \;\middle|\; f_{\mathbf{w},b}(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b) \right\}$$

where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$.

This gives us a hypothesis space that turns our ML task into a parametric optimization problem.

Geometry of Linear Classifiers

The decision function $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$ defines:

- **Decision boundary**: $\{\mathbf{x} : \mathbf{w}^T\mathbf{x} + b = 0\}$ (a hyperplane)
- **Normal vector**: $\mathbf{w}$ is perpendicular to the decision boundary
- **Offset**: b controls the distance from origin

Classification rule:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x} + b > 0 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x} + b < 0 \end{cases}$$

Margin: The signed distance

$$\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|_2}$$

measures confidence. Large margins correspond to robust decisions.

For robust classification we want high confidence and $f(\mathbf{x}_i) = y_i$. By convention, we want $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$ after scaling $(\mathbf{w}, b)$ so $\|\mathbf{w}\|_2 = 1$.

Choosing an Objective: Accuracy?

Natural idea: Maximize 0-1 accuracy

$$\hat{\mathbf{w}}, \hat{b} = \arg\max_{\mathbf{w},b} \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[\text{sign}(\mathbf{w}^T \mathbf{x}_i + b) = y_i]$$

Problems:

1. $J(\mathbf{w}, b)$ is discontinuous with respect to $\mathbf{w}$ and b
2. Cannot use gradient-based optimization!
3. The problem is NP-hard in general

Solution: Replace 0–1 loss with a smooth surrogate that:

- is optimizable,
- encourages large margins.

There are multiple choices of such surrogates!

Definition: For a single example $(\mathbf{x}_i, y_i)$, define the **margin**:

$$m_i = y_i(\mathbf{w}^T\mathbf{x}_i + b)$$

Geometric interpretation: m_i is proportional to the distance of $\mathbf{x}_i$ from the separating hyperplane $\{\mathbf{x} \ : \ \mathbf{w}^T\mathbf{x} + b = 0\}$:

$$m_i = y_i\|\mathbf{w}\|_2 \left(\frac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|_2}\right).$$

Sign of margin:

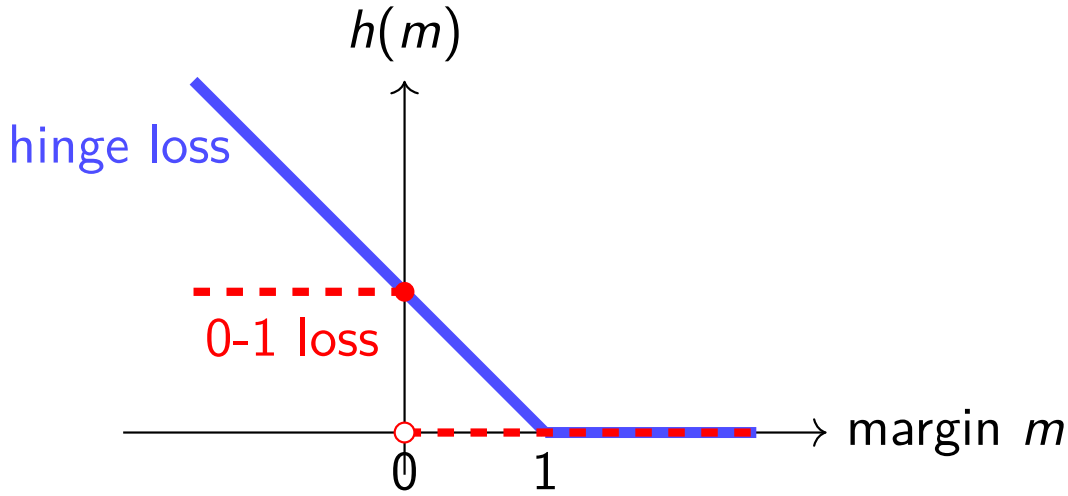
- $m_i > 0$: correct classification
- $m_i < 0$: incorrect classification
- $|m_i|$ large: high confidence

We choose a loss to encourage *large margins* and therefore confidence.

The Hinge Loss

The **hinge loss** is:

$$h(m) = \max(0, 1 - m) = (1 - m)_+ = \begin{cases} 0 & \text{if } m \geq 1 \\ 1 - m & \text{if } m < 1 \end{cases}$$



Takeaway: Hinge loss rewards *correct* predictions that are *confident* (large margin).

Hinge Loss

$$h(m_i) = h(y_i(\mathbf{w}^T\mathbf{x}_i + b)) = (1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+$$

Interpretation:

- If $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$: correct & confident $\Rightarrow$ loss $= 0$
- If $0 < y_i(\mathbf{w}^T\mathbf{x}_i + b) < 1$: correct but not confident $\Rightarrow$ loss > 0
- If $y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0$: incorrect $\Rightarrow$ loss ≥ 1

Average hinge loss on training data:

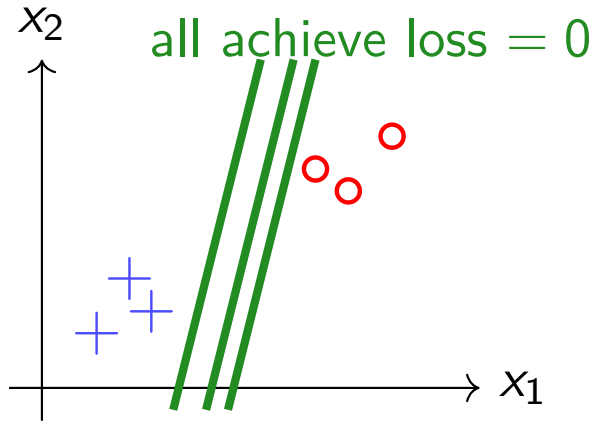
$$J(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+$$

Goal: Minimize $J(\mathbf{w}, b)$ to find a good classifier: minimizing hinge loss implements the idea that "incorrect predictions and small margins should be penalized."

The Indeterminacy Problem

Consider minimizing hinge loss.

Issue: If the data is linearly separable with margin > 1, there are multiple solutions!



Also: If $\alpha \approx 1$ and $(\mathbf{w}, b)$ achieves zero loss, so may $(\alpha\mathbf{w}, \alpha b)$, so there can be infinitely many solutions defining the *same* decision surface!

Implication: Hinge loss alone does not pick a unique model—we need an additional preference. That preference will be expressed through *regularization*.

Breaking Indeterminacy: Regularization

Solution: Add a **regularization term** to prefer simpler models:

$$\hat{\mathbf{w}}, \hat{b} = \arg\min_{\mathbf{w},b} \underbrace{\frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+}_{\text{data-fitting term}} + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|_2^2}_{\text{regularization}}$$

where:

- $\lambda > 0$ is the **regularization parameter** (hyperparameter)
- $\|\mathbf{w}\|_2^2$ is the ℓ_2 **regularizer** or **weight decay**

By varying λ, we can prioritize:

- Fitting the data more (λ small)
- Having a less complex model (λ large)

Effect: Regularization resolves symmetry, controls model complexity, and— as we'll see next— links optimization to generalization.

Why ℓ_2 Regularization?

Geometric view: $\|\mathbf{w}\|_2$ controls the width of the margin. Smaller $\|\mathbf{w}\|_2$ means larger margin.

For points with $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$:

$$\text{distance to boundary} = \frac{|y_i(\mathbf{w}^T\mathbf{x}_i + b)|}{\|\mathbf{w}\|_2} = \frac{1}{\|\mathbf{w}\|_2}$$

$\Rightarrow$ Minimizing $\|\mathbf{w}\|_2^2 \equiv$ Maximizing the margin!

Statistical interpretation: Regularization prevents overfitting

- Small $\|\mathbf{w}\|_2$ means the function changes slowly
- Avoids fitting noise in the training data
- Preview: This will be formalized using probability theory

Other regularizers: ℓ_1 (sparsity), elastic net (combination), etc.

SVM as a Case Study in ERM

The formulation we developed is known as the **soft–margin Support Vector Machine** (SVM). SVM is a *concrete instantiation* of ERM with:

- linear hypothesis space,
- hinge loss,
- ℓ_2 regularization.

SVM Optimization Problem

$$\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+ + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Properties:

- Can be solved efficiently using gradient-based methods
- For some λ, this is equivalent to maximizing the margin subject to allowing some misclassifications

Name origin: The solution depends only on **support vectors** — training points that lie on or inside the margin

Beyond SVMs: General Supervised Learning

The SVM construction illustrates a general ML principle: choose $\mathcal{F}$, choose ℓ, collect $\mathcal{D}$, and minimize empirical risk.

Components:

- **Loss function** $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$
  Measures the cost of predicting $\hat{y}$ when the true value is y
- **Hypothesis space** $\mathcal{F}$ — the class of candidate models f
- **Training data** $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$

ERM: Find $f^* \in \mathcal{F}$ that minimizes:

$$f^* \in \arg\min_{f \in \mathcal{F}} \hat{R}(f) := \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i)}_{\text{empirical risk}}$$

Examples of Loss Functions

Regression (continuous $y \in \mathbb{R}$):

- Squared loss: $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$
- Absolute loss: $\ell(\hat{y}, y) = |\hat{y} - y|$
- Huber loss: smooth combination of squared and absolute

Binary classification ($y \in \{-1, +1\}$):

- Hinge loss: $\ell(f(\mathbf{x}), y) = (1 - yf(\mathbf{x}))_+$
- Logistic loss: $\ell(f(\mathbf{x}), y) = \log(1 + e^{-yf(\mathbf{x})})$
- Exponential loss: $\ell(f(\mathbf{x}), y) = e^{-yf(\mathbf{x})}$

Multi-class classification ($y \in \{1, \ldots, K\}$):

- Cross-entropy loss (coming in later lectures)

Examples of Hypothesis Spaces

Linear models:

$$\mathcal{F}_{\text{linear}} = \{f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

Polynomial models:

$$\mathcal{F}_{\text{poly}} = \{f_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{|\boldsymbol{\alpha}| \leq k} \theta_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}\}$$

Kernel methods:

$$\mathcal{F}_{\text{kernel}} = \left\{ f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) \right\}$$

Neural networks:

$$\mathcal{F}_{\text{NN}} = \{\text{compositions of affine maps and nonlinearities}\}$$

Each choice of $\mathcal{F}$ embodies different **inductive biases** about the problem

ERM with Regularization

In practice, we often add regularization:

$$f^* \in \arg\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i)}_{\text{empirical risk}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}}$$

where $\Omega(f)$ measures the complexity of f

Common regularizers:

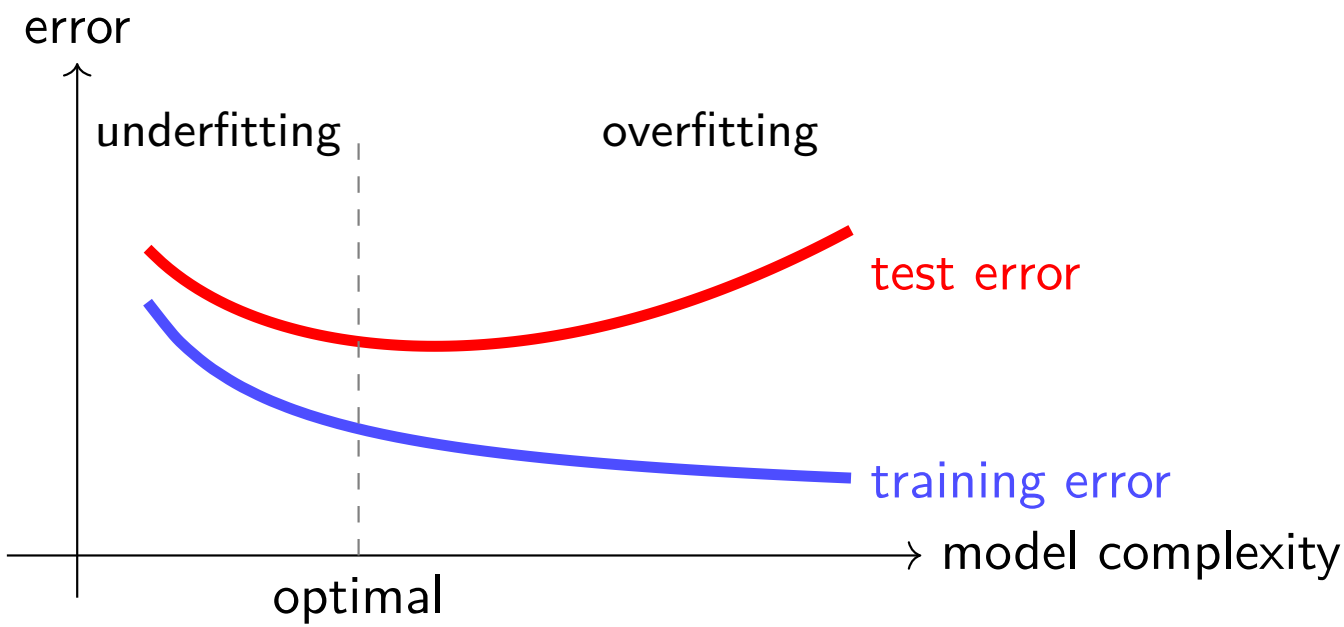
- ℓ_2: $\Omega(f) = \|\mathbf{w}\|_2^2$ (ridge, weight decay)
- ℓ_1: $\Omega(f) = \|\mathbf{w}\|_1$ (LASSO, sparsity)
- Elastic net: $\Omega(f) = \alpha\|\mathbf{w}\|_1 + (1-\alpha)\|\mathbf{w}\|_2^2$
- Dropout, batch normalization (for neural networks)

λ is a **hyperparameter** chosen via validation

The Key Question: Generalization

Training objective: Minimize empirical risk $\hat{R}(f)$ on training data

True goal: Perform well on *new*, *unseen* data!



Problem: Our data $\{(\mathbf{x}_i, y_i)\}$ is *random*!
Key puzzle: When does a solution to ERM also perform well on unseen data?

Preview: Population Risk

Key insight: Training data is sampled from an unknown distribution $\mathbb{P}$

The **population risk** (or **true risk**) is:

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

This is what we *really* want to minimize!

Problem: We don't know $\mathbb{P}$, so we can't compute $R(f)$, much less optimize. Our optimization was on training data. Learning theory asks (and answers) when this matches performance on $\mathbb{P}$.

ERM approximation: $\hat{R}(f) \approx R(f)$ when n is large

Next lecture: We'll formalize this using probability theory:

- What does $\mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}}[\cdot]$ mean?
- When is $\hat{R}(f)$ close to $R(f)$?
- How does this depend on n, $\mathcal{F}$, and ℓ?

Summary

Key concepts:

1. Machine learning = finding predictable patterns from random data
2. Hypothesis space $\mathcal{F}$ encodes our assumptions about the problem
3. Loss function ℓ measures prediction quality
4. Empirical risk minimization: $\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i)$
5. Regularization prevents overfitting by penalizing complexity

SVM served as a worked example of ERM: it shows how geometry, optimization, and generalization interconnect. We developed the soft SVM objective

$$\min_{\mathbf{w},b} \frac{1}{n} \sum_{i=1}^{n} (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))_+ + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

Next time: Probability theory foundations

- Populations risk vs empirical risk and justification of ERM via LLN
- Parametric modeling of distributions for principled statistical learning from finite data
- Maximum Likelihood (MLE) and Maximum A Posteriori (MAP) Estimation of parameters