Machine Learning and Optimization

Lecture 3: Probabilistic Foundations and Regression Models

CSCI 4140/6960 —Spring 2026

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

Lecture 3 Roadmap

The Generalization Problem

Probability as a Tool for Reasoning About Prediction

Parametric Families: Assumptions Enable Learning

The Gaussian Family for Regression

Maximum Likelihood: From Models to Loss Functions

Bayesian Inference: The Justification for Regularization

Summary and Looking Forward

Recall: ERM Solves an Optimization Problem

Last lecture: We formulated supervised learning as

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

We developed this for linear regression and SVM:

- Regression: minimize squared error $(y_i - \mathbf{w}^T \mathbf{x}_i)^2$
- SVM: minimize hinge loss $(1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))_+$
- Both included regularization: $\frac{\lambda}{2} \|\mathbf{w}\|_2^2$

These were motivated by intuitive arguments

- Squared loss: penalizes large prediction errors
- Hinge loss: encourages correct classification with margin
- Regularization: prevents overfitting (but why this form?)

Today: derive these models from statistical principles

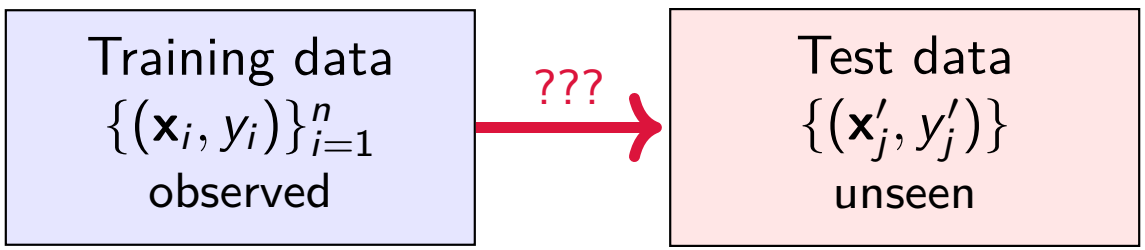
The Central Tension in Machine Learning

ERM fits the past.

How can we trust it on the future?

The problem:

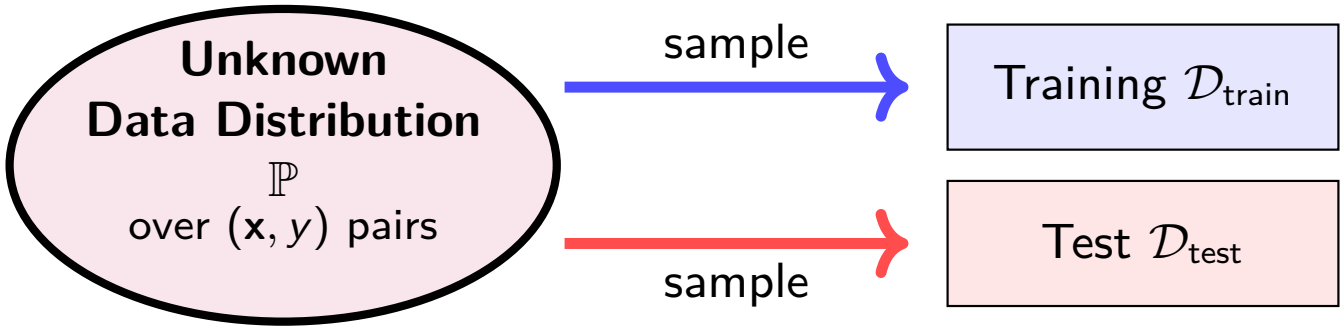
- We minimize loss on n training examples
- But we care about performance on *new, unseen* test data
- Training data is finite and random
- Test data is unknown at training time



Today's goal: Understand how **probability theory** gives us the language to reason about this problem

Why Probability is Necessary

Observation: Training and test data are both **random samples**



Key insight: Both datasets come from the *same* unknown distribution $\mathbb{P}$

This means:

- A model that fits training data well *might* generalize well
- But only if training data is **representative** of the true distribution
- Probability theory lets us formalize "representative" and "generalize"

Without probability, we have no formal way to reason about generalization!

From Empirical Risk to True Risk

What we compute (empirical risk):

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i)$$

Average loss on the n training examples we happened to observe

What we actually care about (population risk):

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

Expected loss over *all possible* examples drawn from $\mathbb{P}$



The learning theory question: When is $\hat{R}(f) \approx R(f)$?

Roadmap for Today

To understand when ERM generalizes, we need to:

1. **Define the probabilistic framework**
   - What does $\mathbb{E}[\cdot]$ mean?
   - How do we model $\mathbb{P}$?

2. **Restrict to parametric families**
   - Can't learn arbitrary distributions from finite data
   - Assume $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$ for known family, unknown $\boldsymbol{\theta}$

3. **Connect modeling to optimization**
   - Maximum Likelihood: where squared loss comes from
   - Bayesian inference: where regularization comes from

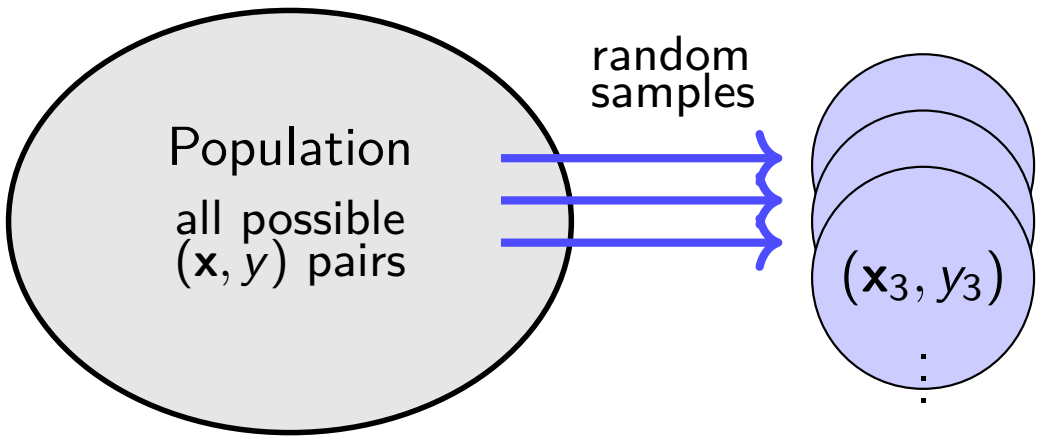
Goal for today: By the end, you'll see how probability *explains* the regression models and regularization from previous lectures

Random Variables: Formalizing Uncertainty

Definition: A **random variable** is a quantity whose value is determined by a random process

In ML:

- X: input features (e.g., pixel values, patient measurements)
- Y: output label (e.g., house price, temperature, class label)
- Both are random because they're *sampled* from a population

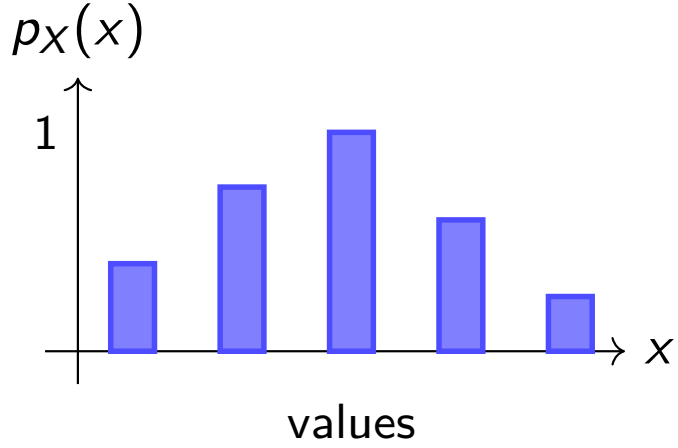


Key point: We don't control which $(\mathbf{x}_i, y_i)$ we get—they're random! This randomness is *why* we need probability to reason about learning

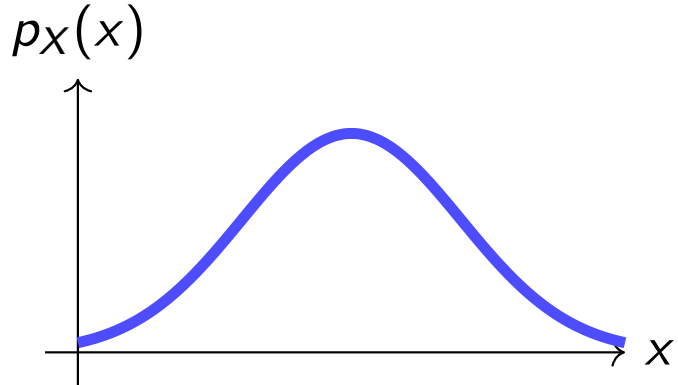
Distributions: Encoding What's Likely

A **probability distribution** $\mathbb{P}$ describes how likely different outcomes are.

Discrete case: $X \in \{x_1, x_2, \ldots\}$
Probability mass function (PMF): $p_X(x) = \mathbb{P}(X = x)$



Continuous case: $X \in \mathbb{R}^d$
Probability density function (PDF): $\mathbb{P}(X \in A) = \int_A p_X(\mathbf{x})\, d\mathbf{x}$



Expectation: The Average Outcome

The **expectation** $\mathbb{E}[X]$ is the *average value* if we sampled infinitely many times

$$\mathbb{E}[X] = \begin{cases} \sum_x x \cdot p_X(x) & \text{discrete} \\ \int x \cdot p_X(x)\, dx & \text{continuous} \end{cases}$$

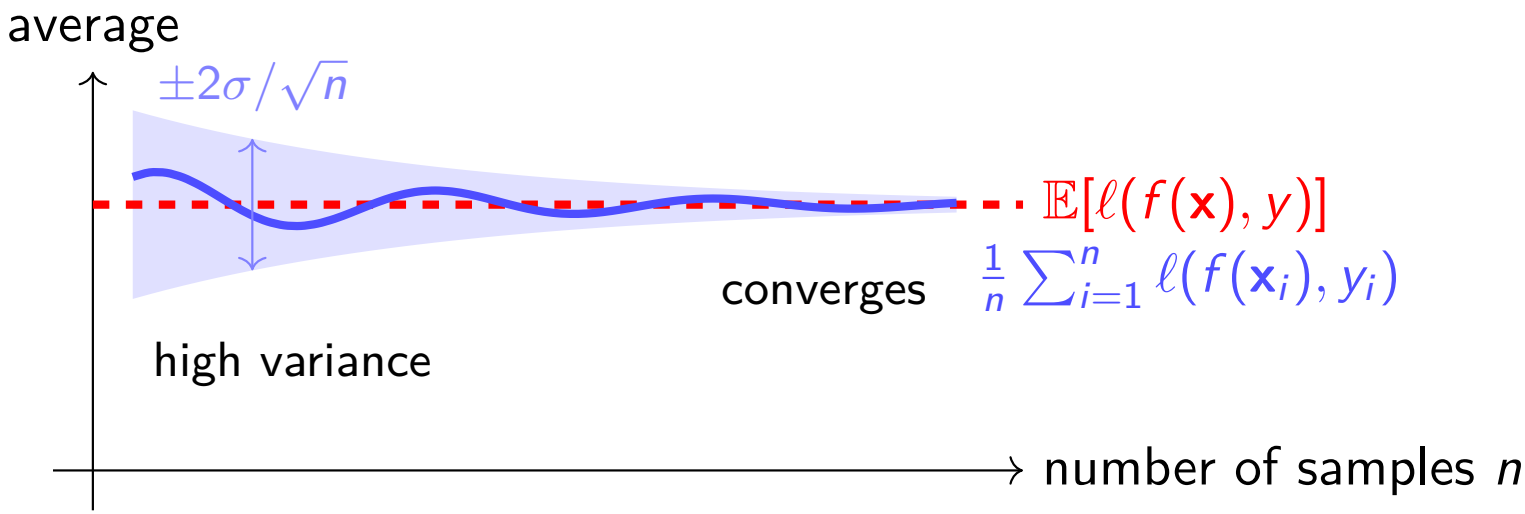
Why this matters for ML:
The population risk $R(f) = \mathbb{E}[\ell(f(\mathbf{x}), y)]$ is the **expected loss**

- Not the loss on one example
- Not the loss on our n training examples
- The *average* loss over all possible future examples

Connection: Minimizing $R(f)$ means finding f with best *average-case* performance

Law of Large Numbers: Why ERM Works

Intuition: Sample averages converge to expectations as $n \to \infty$



Formal statement (LLN):

$$\frac{1}{n}\sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) \xrightarrow{n\to\infty} \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}}[\ell(f(\mathbf{x}), y)]$$

Implication: For large n, empirical risk $\hat{R}(f)$ approximates population risk $R(f)$

This is why ERM can work! But we need enough data and the right model class $\mathcal{F}$

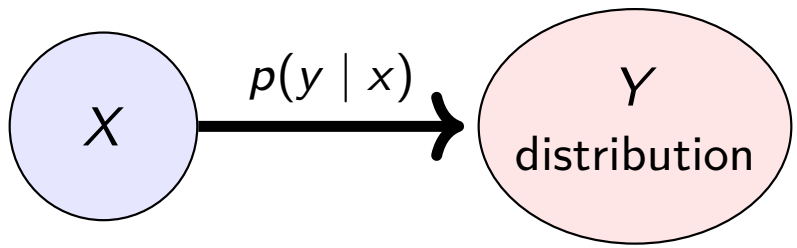
Conditional Distributions: Modeling Prediction

The supervised learning setup: We want to predict Y from X

The **conditional distribution** $p_{Y|X}(y \mid x)$ encodes this:

$$p_{Y|X}(y \mid x) = \frac{p_{X,Y}(x, y)}{p_X(x)}$$

"Given that we observe $X = x$, what's the distribution over Y?"



Different x values
give different Y distributions

ML connection:

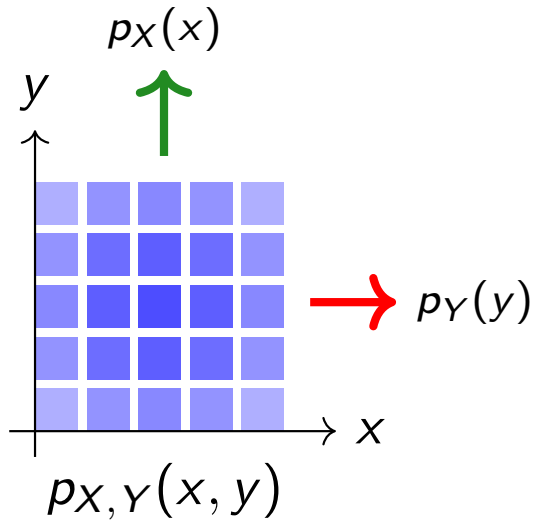
- Regression: $p(y \mid \mathbf{x})$ is typically Gaussian centered at prediction
- Classification: $p(y \mid \mathbf{x})$ gives probabilities for different classes

Joint and Marginal Distributions

Joint distribution: $p_{X,Y}(x, y)$ describes both variables together

Marginal distribution: Recover $p_X(x)$ by "summing out" Y:

$$p_X(x) = \sum_y p_{X,Y}(x, y) \quad \text{or} \quad p_X(x) = \int p_{X,Y}(x, y)\, dy$$



ML connection: In supervised learning:

- $p_{X,Y}(x, y) = p_{Y|X}(y \mid x) \cdot p_X(x)$ (generative approach)
- We model $p(y \mid x)$ but don't need to model $p_X(x)$ (discriminative approach)

Independence and the i.i.d. Assumption

Random variables X and Y are **independent** (written $X \perp\!\!\!\perp Y$) if:

$$p_{X,Y}(x, y) = p_X(x) \cdot p_Y(y)$$

Knowing X tells you nothing about Y

The i.i.d. assumption in ML:
Training examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ are **independent and identically distributed**:

- **Independent:** Observing one doesn't affect others
- **Identically distributed:** All drawn from same $\mathbb{P}$

Why this matters:

$$p(\mathcal{D}) = p((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)) = \prod_{i=1}^{n} p(\mathbf{x}_i, y_i)$$

Independence lets us *factor* the joint probability $\Rightarrow$ tractable learning algorithms

Caveat: i.i.d. assumption can fail (time series, medical data from same patient, etc.)

How Probability Defines Learning Goals

What we've established:

1. **Data is random:** $(\mathbf{x}, y) \sim \mathbb{P}$ for unknown $\mathbb{P}$
2. **Goal is expected performance:**

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

3. **ERM approximates this:**

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) \approx R(f)$$

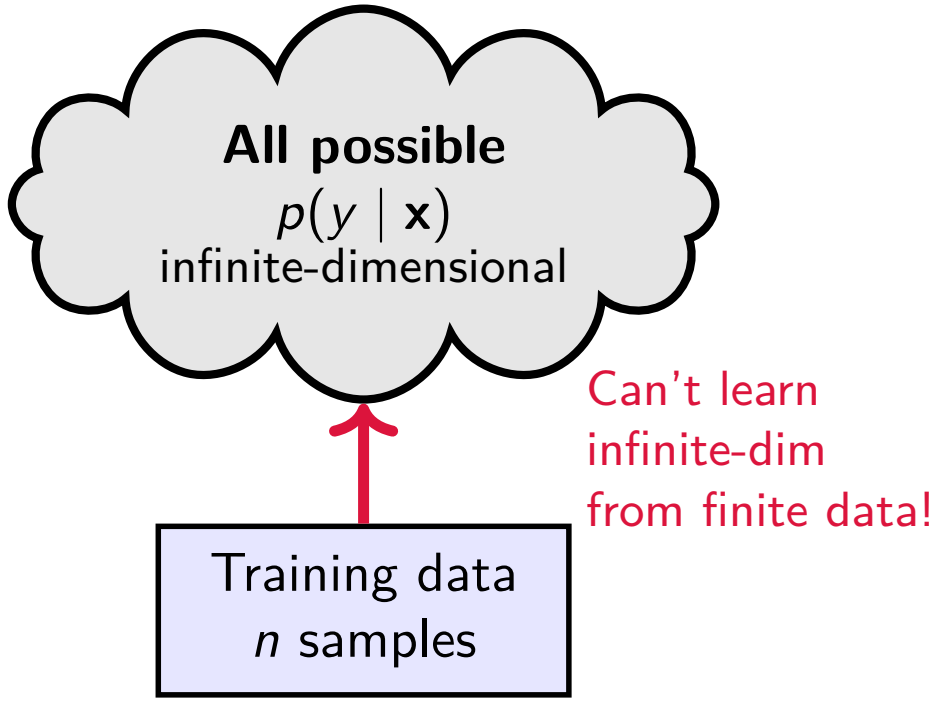
   when n is large (Law of Large Numbers)

4. **i.i.d. assumption:** Lets us reason about samples independently

Key insight: Probability converts the vague idea of "generalization" into the concrete mathematical question: "When is $\hat{R}(f) \approx R(f)$?"

The Impossibility of Learning Arbitrary Distributions

Problem: The space of all possible distributions is infinite-dimensional



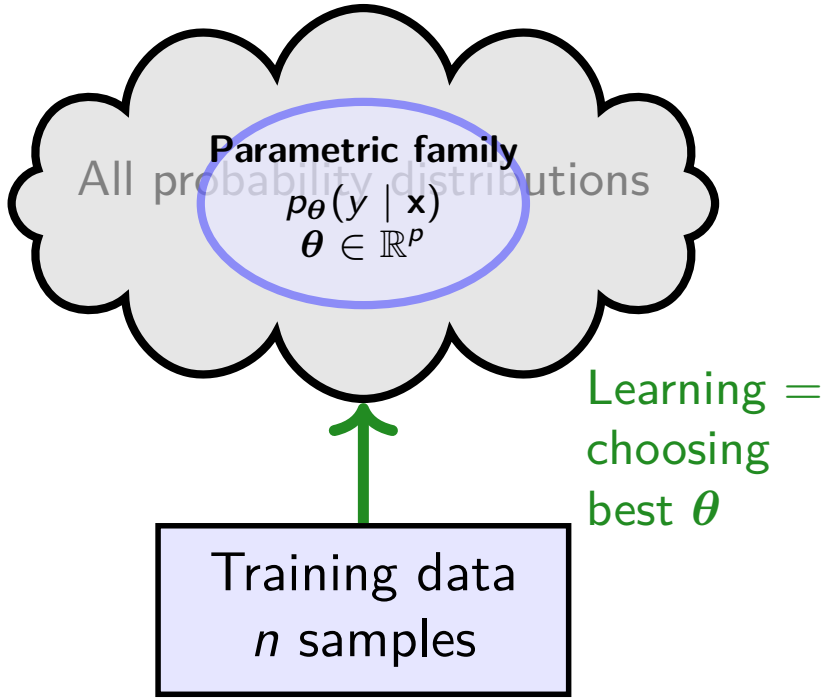
Observation: With n examples, we can only learn $\approx n$ parameters

Solution: Make **parametric assumptions**

- Restrict to a *family* of distributions $\{p_{\boldsymbol{\theta}}(y \mid \mathbf{x}) : \boldsymbol{\theta} \in \Theta \subset \mathbb{R}^p\}$
- Learning = finding the right $\boldsymbol{\theta}$ (finite-dimensional!)

Parametric Models: Trading Flexibility for Tractability

Parametric family: Assume $p(y \mid \mathbf{x})$ has a known functional form with unknown parameters $\boldsymbol{\theta}$



The trade-off:

- **Gain:** Learning becomes tractable (optimize over $\mathbb{R}^p$)
- **Loss:** True $p(y \mid \mathbf{x})$ might not be in our family (**model misspecification**)

Key modeling choice: Pick family that balances expressiveness and simplicity

Why Parametric Assumptions Enable Generalization

Informal argument:

1. Without assumptions, we need to estimate $p(y \mid \mathbf{x})$ for *every possible* $\mathbf{x}$

   $\Rightarrow$ Need infinite data (or see same $\mathbf{x}$ many times)

2. With parametric assumptions, we estimate p *parameters*

   $\Rightarrow$ Need $n \gtrsim p$ examples (roughly)

3. Parameters **share statistical strength** across examples:
   - Data at $\mathbf{x}_1$ informs predictions at nearby $\mathbf{x}_2$
   - Because they use the *same* $\boldsymbol{\theta}$

Bottom line: Parametric modeling is what makes learning from finite data possible

Next: What parametric family should we use for regression?

Gaussian Model: Linear Regression with Noise

Task: Predict continuous $y \in \mathbb{R}$ from $\mathbf{x} \in \mathbb{R}^d$

Assumption: Observations are affine functions of $\mathbf{x}$ corrupted by Gaussian noise

$$y = \mathbf{w}^T \mathbf{x} + b + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

Equivalently:

$$y \mid \mathbf{x} \sim \mathcal{N}(\mathbf{w}^T \mathbf{x} + b, \sigma^2)$$

with density:

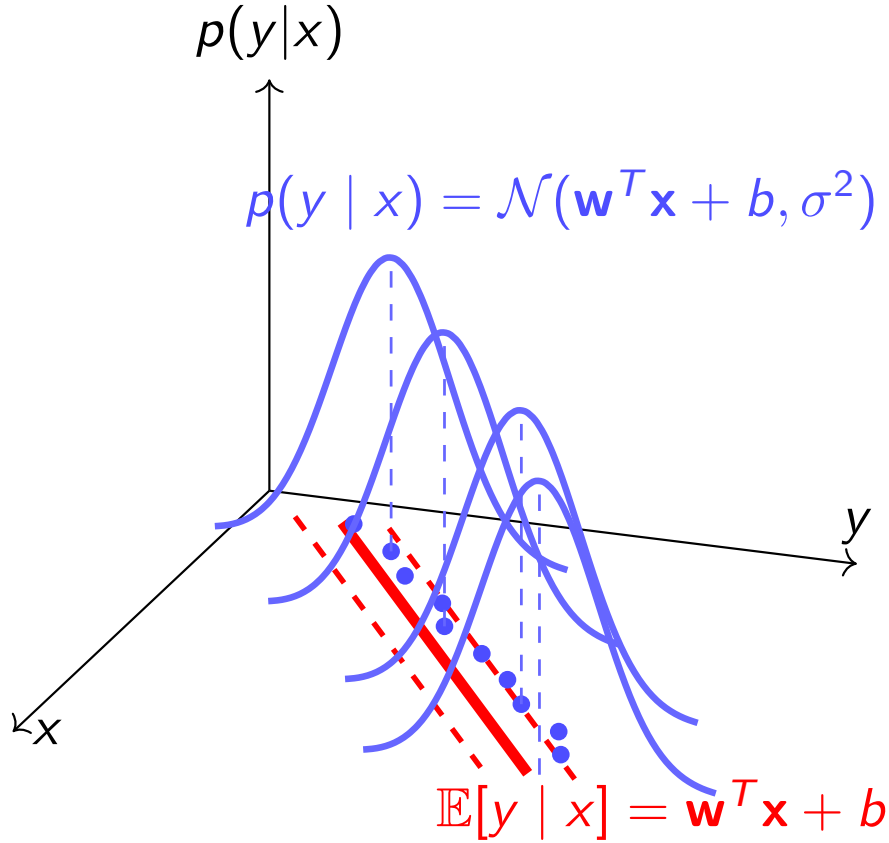
$$p_{\boldsymbol{\theta}}(y \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mathbf{w}^T \mathbf{x} - b)^2}{2\sigma^2}\right)$$

Parameters: $\boldsymbol{\theta} = (\mathbf{w}, b, \sigma^2)$

Interpretation:

- $\mathbb{E}[y \mid \mathbf{x}] = \mathbf{w}^T \mathbf{x} + b$ (mean prediction)
- $\mathrm{Var}(y \mid \mathbf{x}) = \sigma^2$ (uncertainty)

Visualizing the Gaussian Regression Model



Key insight: Each $\mathbf{x}$ gives a *distribution* over y, not a single value.

This models irreducible uncertainty (measurement noise, unobserved factors, etc.)

Maximum Likelihood Estimation (MLE)

Setup: Given data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and model $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$

MLE principle: Choose $\boldsymbol{\theta}$ that makes the observed data most likely

Likelihood: Probability of data given parameters (assuming i.i.d.)

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(y_i \mid \mathbf{x}_i)$$

Log-likelihood: Take log for numerical stability

$$\ell(\boldsymbol{\theta}) = \log \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(y_i \mid \mathbf{x}_i)$$

MLE: Maximize log-likelihood (equivalently, minimize negative log-likelihood)

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \underbrace{-\frac{1}{n} \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(y_i \mid \mathbf{x}_i)}_{\text{NLL (negative log-likelihood)}}$$

Example: From Gaussian to Squared Loss

Model: $y \mid \mathbf{x} \sim \mathcal{N}(\mathbf{w}^T\mathbf{x} + b, \sigma^2)$

Negative log-likelihood:

$$
\begin{aligned}
-\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x}) &= -\log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y - \mathbf{w}^T\mathbf{x} - b)^2}{2\sigma^2}\right)\right] \\
&= \frac{(y - \mathbf{w}^T\mathbf{x} - b)^2}{2\sigma^2} + \underbrace{\log(\sqrt{2\pi\sigma^2})}_{\text{constant}^\dagger \text{ w.r.t. } \mathbf{w}} \\
&\propto (y - \mathbf{w}^T\mathbf{x} - b)^2
\end{aligned}
$$

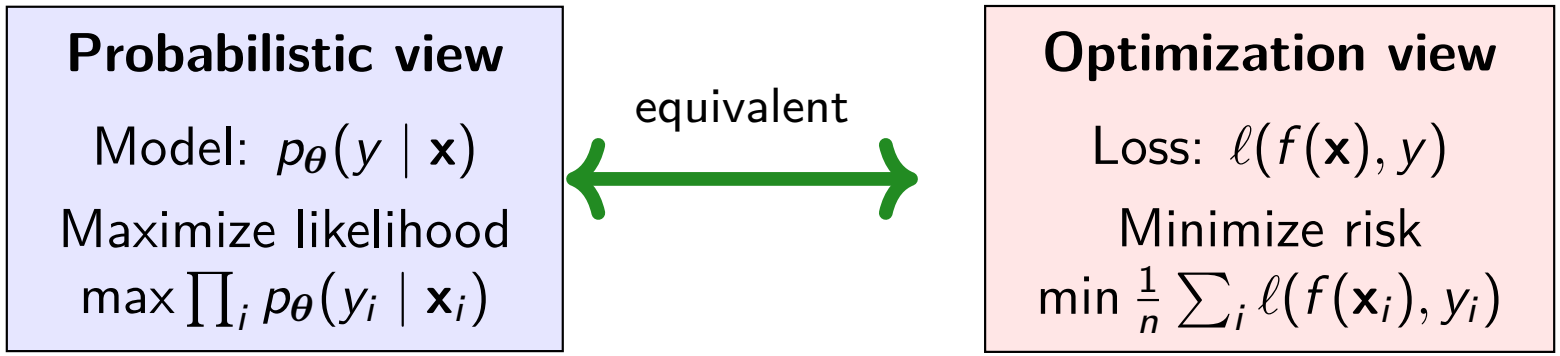
Result: Gaussian MLE gives **squared loss**!

> This explains why least squares regression works:
> It's the MLE under Gaussian noise assumption

† we assume σ^2 is known to get a nice (convex) optimization problem that does not involve σ^2.

MLE = ERM with NLL Loss

Key observation: Maximizing likelihood is equivalent to minimizing a loss!



Define loss as: $\ell(f(\mathbf{x}), y) = -\log p_{\theta}(y \mid \mathbf{x})$
Then:

$$\text{MLE} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \underbrace{(-\log p_{\theta}(y_i \mid \mathbf{x}_i))}_{\text{loss function}}$$

This is exactly ERM! Different models give different loss functions.
When $p_{\theta}(y \mid \mathbf{x})$ is Gaussian,

$$\text{MLE} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} -\log p_{\theta}(y_i \mid \mathbf{x}_i) = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \underbrace{|y_i - \mathbf{w}^T \mathbf{x} - b|^2}_{\ell_2 \text{ loss function}}$$

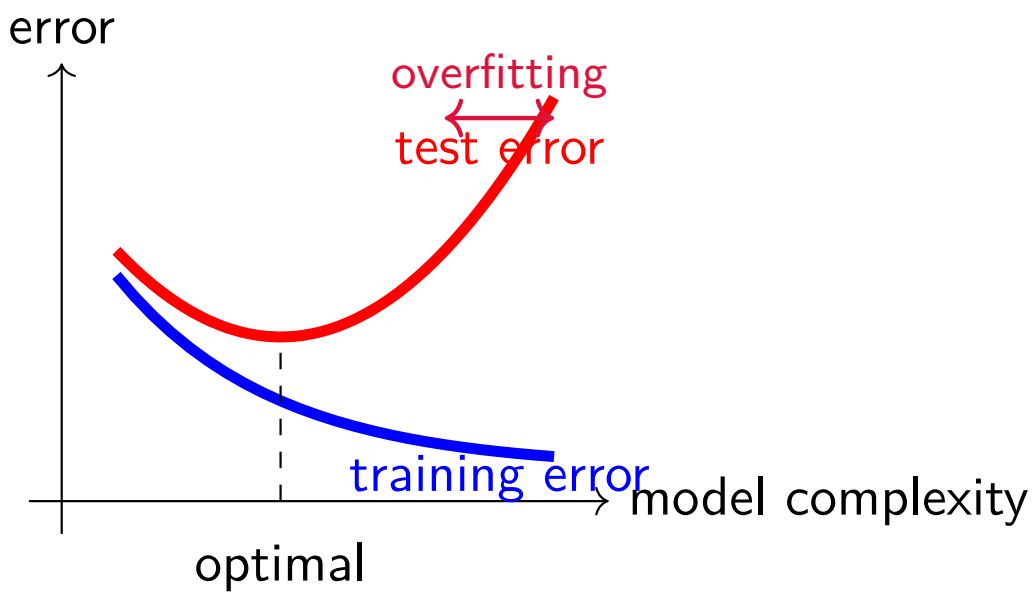
The Problem with MLE: Overfitting

MLE can overfit when:

- Data is limited (n small)
- Model is complex (p large, where $p = \dim(\boldsymbol{\theta})$)
- Some parameters are poorly constrained by data

Example: High-dimensional linear regression ($d > n$)

- MLE has infinitely many solutions
- Can perfectly fit training data (zero training error)
- But generalizes poorly to test data



We need a principled way to prefer simpler models!

Enter Bayes: Parameters are Uncertain Too

Key insight: Parameters $\boldsymbol{\theta}$ are also uncertain! Use probability to model this uncertainty.

Bayesian framework:

1. **Prior distribution** $p(\boldsymbol{\theta})$: Our beliefs about $\boldsymbol{\theta}$ *before* seeing data
   - Encodes preference for certain parameter values
   - E.g., prefer small weights: $\mathbf{w} \approx \mathbf{0}$
2. **Likelihood** $p(\mathcal{D} \mid \boldsymbol{\theta})$: Probability of data given parameters
   - Same as in MLE: $\prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta})$
3. **Posterior distribution** $p(\boldsymbol{\theta} \mid \mathcal{D})$: Our beliefs about $\boldsymbol{\theta}$ *after* seeing data

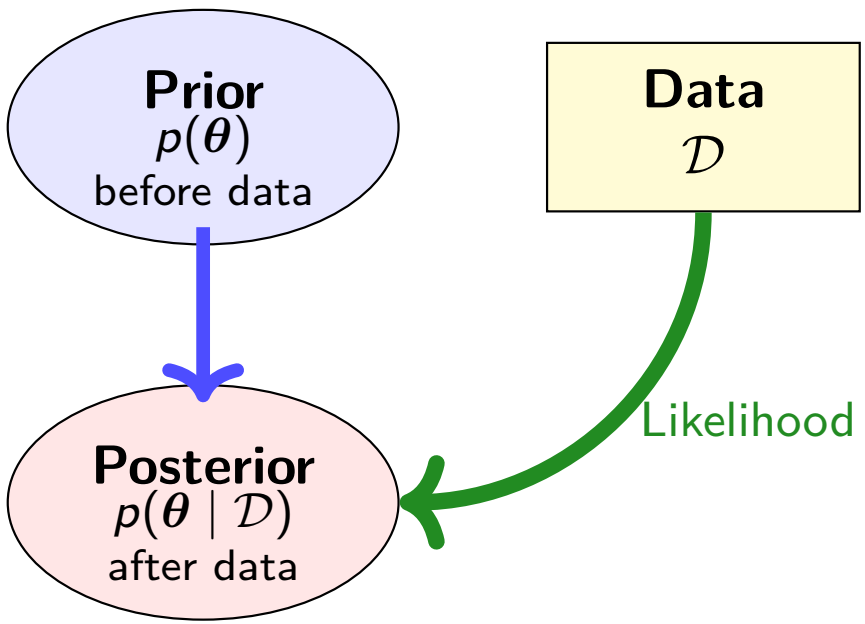
Bayes' rule:

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})}{p(\mathcal{D})}$$

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$

Bayes' Rule: Updating Beliefs with Data

$$\underbrace{p(\boldsymbol{\theta} \mid \mathcal{D})}_{\text{posterior}} = \frac{\overbrace{p(\mathcal{D} \mid \boldsymbol{\theta})}^{\text{likelihood}} \cdot \overbrace{p(\boldsymbol{\theta})}^{\text{prior}}}{\underbrace{p(\mathcal{D})}_{\text{evidence}}}$$



Interpretation: Prior beliefs + Data evidence $\Rightarrow$ Updated beliefs (posterior)

Maximum A Posteriori (MAP) Estimation

Full Bayesian inference: Use entire posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$
This is computationally expensive (need to integrate over all $\boldsymbol{\theta}$)

MAP estimation: Find the *single most likely* value of $\boldsymbol{\theta}$ under the posterior

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D})$$

Using Bayes' rule:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{MAP}} &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) \\
&= \arg\max_{\boldsymbol{\theta}} \frac{p(\mathcal{D} \mid \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})}{p(\mathcal{D})} \\
&= \arg\max_{\boldsymbol{\theta}} p(\mathcal{D} \mid \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}) \quad (p(\mathcal{D}) \text{ doesn't depend on } \boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} [\log p(\mathcal{D} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})]
\end{aligned}$$

Takeaway: MAP = MLE + Prior term!

MAP as Regularized MLE

MAP objective (taking log):

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} \underbrace{\sum_{i=1}^{n} \log p(y_i \mid \mathbf{x}_i, \theta)}_{\text{log-likelihood (data fit)}} + \underbrace{\log p(\theta)}_{\text{log-prior (regularization)}}$$

Equivalently, minimize the negative:

$$\hat{\theta}_{\text{MAP}} = \arg\min_{\theta} \underbrace{-\sum_{i=1}^{n} \log p(y_i \mid \mathbf{x}_i, \theta)}_{\text{NLL loss}} + \underbrace{(-\log p(\theta))}_{\text{regularization penalty}}$$

MLE fit data	+	Prior prefer simple θ	=	MAP **regularized fit**

Takeaway: The prior acts as a **regularization term** that penalizes complex models!

Example: Gaussian Prior Gives Ridge Regression

Model: $y \mid \mathbf{x}, \mathbf{w} \sim \mathcal{N}(\mathbf{w}^T\mathbf{x}, \sigma^2)$

Prior on weights: Assume $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \tau^2 I)$

$$p(\mathbf{w}) = \frac{1}{(2\pi\tau^2)^{d/2}} \exp\left(-\frac{\|\mathbf{w}\|_2^2}{2\tau^2}\right)$$

This encodes the belief that weights should be *small* (close to zero)

MAP objective:

$$\begin{aligned}\hat{\mathbf{w}}_{\text{MAP}} &= \arg\max_{\mathbf{w}} \left[\log p(\mathcal{D} \mid \mathbf{w}) + \log p(\mathbf{w})\right] \\ &= \arg\max_{\mathbf{w}} \left[\sum_{i=1}^{n} \log p(y_i \mid \mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w})\right]\end{aligned}$$

Let's compute each term...

Deriving Ridge Regression from MAP

Log-likelihood term:

$$\sum_{i=1}^{n} \log p(y_i \mid \mathbf{x}_i, \mathbf{w}) = \sum_{i=1}^{n} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{w}^T \mathbf{x}_i)^2}{2\sigma^2} \right) \right]$$

$$= -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \text{const}$$

Log-prior term:

$$\log p(\mathbf{w}) = \log \left[\frac{1}{(2\pi\tau^2)^{d/2}} \exp\left(-\frac{\|\mathbf{w}\|_2^2}{2\tau^2} \right) \right]$$

$$= -\frac{1}{2\tau^2} \|\mathbf{w}\|_2^2 + \text{const}$$

Combined (dropping constants):

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\max_{\mathbf{w}} \left[-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 - \frac{1}{2\tau^2} \|\mathbf{w}\|_2^2 \right]$$

Ridge Regression = MAP with Gaussian Prior

Equivalently (minimize negative):

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\min_{\mathbf{w}} \left[\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \frac{1}{2\tau^2} \|\mathbf{w}\|_2^2 \right]$$

Multiply through by $2\sigma^2$ and set $\lambda = \sigma^2/\tau^2$:

$$\hat{\mathbf{w}}_{\text{ridge}} = \arg\min_{\mathbf{w}} \left[\sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2 \right]$$

This is exactly ridge regression!

The ℓ_2 regularization term comes from assuming a Gaussian prior on weights

Interpretation of λ:

- $\lambda = \sigma^2/\tau^2 = \frac{\text{noise variance}}{\text{prior variance}}$
- Large λ: Strong belief that weights should be small
- Small λ: Weak prior, mostly trust the data

Why Ridge Regression Works

From a Bayesian perspective:

1. The Gaussian prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \tau^2 I)$ encodes **inductive bias**:
   - We expect most features to have small effects
   - Extreme parameter values are unlikely
2. When data is limited, the prior **regularizes** the solution:
   - Prevents overfitting by shrinking weights toward zero
   - Trades off bias (from prior) vs. variance (from data)
3. The strength of regularization (λ) controls this trade-off:
   - High λ: Strong regularization, simpler model, higher bias, lower variance
   - Low λ: Weak regularization, more complex model, lower bias, higher variance

Bottom line: Ridge regression is not ad hoc—it's the principled Bayesian solution under a Gaussian prior

Other Priors Give Other Regularizers

General principle: Different prior distributions $p(\theta)$ give different regularization penalties

Prior	**Form**	**Regularizer**
Gaussian	$p(\mathbf{w}) \propto \exp\left(-\frac{\|\mathbf{w}\|_2^2}{2\tau^2}\right)$	$\lambda\|\mathbf{w}\|_2^2$ (Ridge)
Laplace	$p(\mathbf{w}) \propto \exp\left(-\frac{\|\mathbf{w}\|_1}{\tau}\right)$	$\lambda\|\mathbf{w}\|_1$ (Lasso)
Uniform	$p(\mathbf{w}) \propto \mathbf{1}_{\|\mathbf{w}\|_2 \leq R}$	Hard constraint $\|\mathbf{w}\|_2 \leq R$

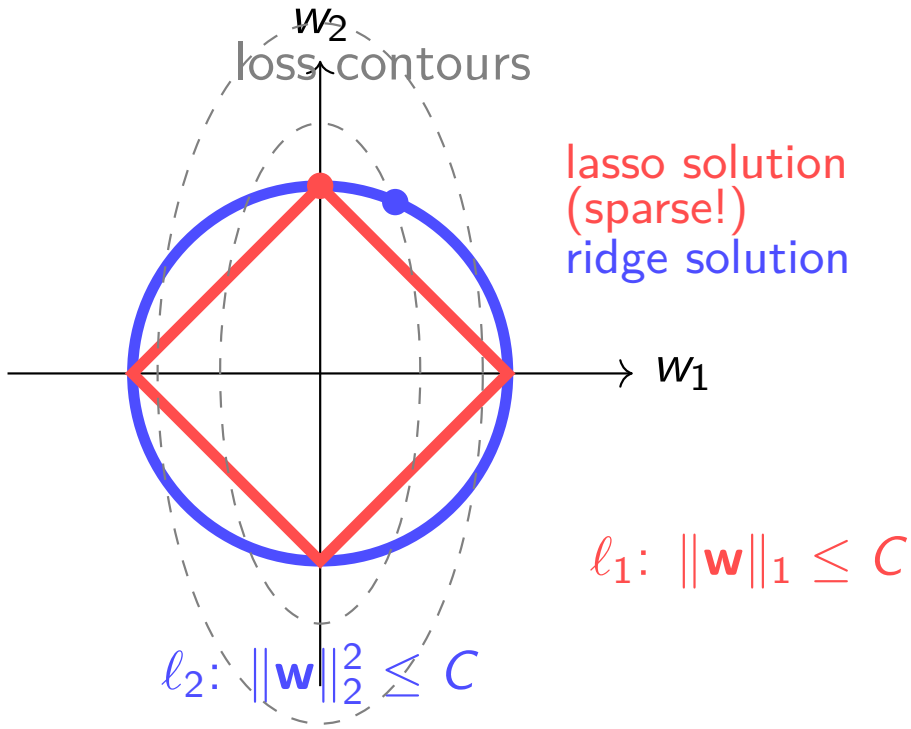
Key insight:

- Gaussian prior $\Rightarrow$ quadratic penalty $\Rightarrow$ shrinks weights smoothly
- Laplace prior $\Rightarrow$ absolute value penalty $\Rightarrow$ promotes sparsity (many weights $= 0$)

Bayesian framework unifies many regularization techniques!

Geometric Interpretation: Constraint on Parameters

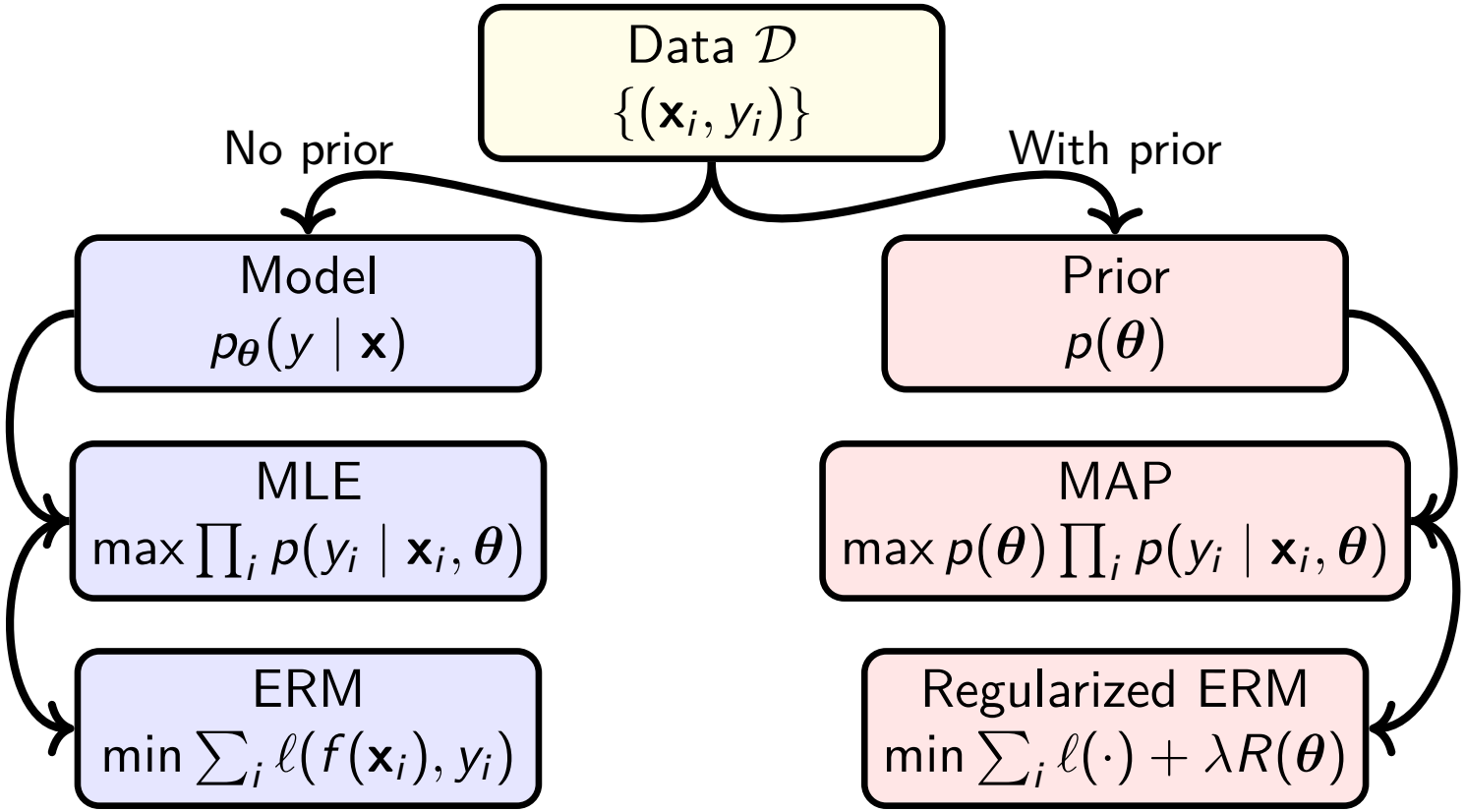
Alternative view: Regularization constrains the parameter space



Key difference:

- Ridge (ℓ_2): Circular constraint $\Rightarrow$ solution typically has all non-zero weights
- Lasso (ℓ_1): Diamond constraint $\Rightarrow$ solution often has many zero weights (sparsity)

Summary: The Complete Statistical Picture



Key insights:

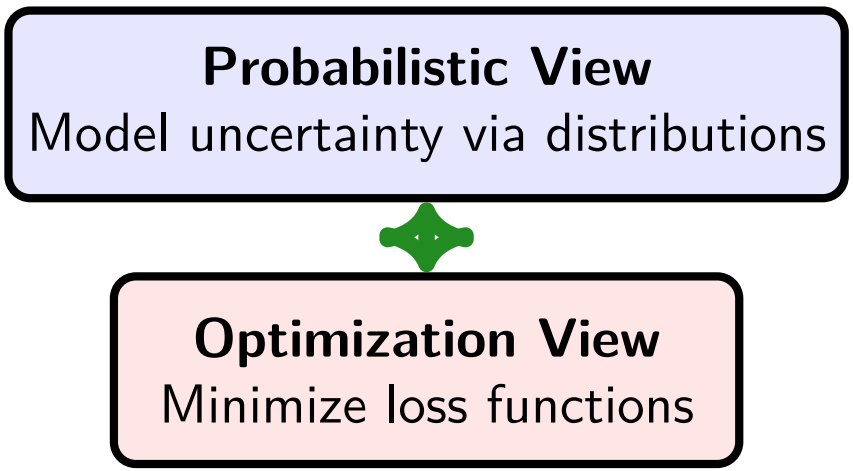
1. MLE $\Leftrightarrow$ ERM with NLL loss (e.g., squared loss for Gaussian)
2. MAP = MLE + Prior $\Leftrightarrow$ Regularized ERM
3. Gaussian prior $\Rightarrow$ Ridge regression (ℓ_2 regularization)

What We've Established Today

1. **Probabilistic foundations:**
   - Data is random samples from unknown distribution $\mathbb{P}$
   - Goal: minimize expected risk $R(f) = \mathbb{E}[\ell(f(\mathbf{x}), y)]$
   - ERM approximates this via LLN when n is large
2. **Parametric modeling:**
   - Restrict to family $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$ to enable learning from finite data
   - Gaussian model: $y \mid \mathbf{x} \sim \mathcal{N}(\mathbf{w}^T\mathbf{x}, \sigma^2)$
3. **Maximum likelihood:**
   - MLE $\Leftrightarrow$ ERM with negative log-likelihood loss
   - Gaussian MLE $\Rightarrow$ least squares regression
4. **Bayesian perspective on regularization:**
   - Prior distribution $p(\boldsymbol{\theta})$ encodes preferences
   - MAP = MLE + Prior = Regularized ERM
   - Gaussian prior $\Rightarrow$ Ridge regression (ℓ_2 regularization)

We've now given principled statistical justifications for the ad hoc methods from previous lectures!

The Big Picture: Two Views, One Framework



- Model: $p_\theta(y \mid \mathbf{x})$
- MLE: maximize likelihood
- MAP: add prior
- Interpretable predictions

- Loss: $\ell(f(\mathbf{x}), y)$
- ERM: minimize empirical risk
- Add regularization
- Computational algorithms

Unified message: Probability provides the *why*, optimization provides the *how*. Together they give us principled, practical machine learning.

Next Lecture: Classification Models

Today: We derived regression models from statistical first principles

- Gaussian likelihood $\Rightarrow$ squared loss
- Gaussian prior $\Rightarrow$ ridge regularization

Lecture 4 will cover:

1. **Probabilistic models for classification**
   - Bernoulli distribution for binary classification
   - Categorical distribution for multi-class problems
   - Softmax function and its properties
2. **MLE for classification**
   - Deriving logistic loss and cross-entropy loss
   - Connection to logistic regression
3. **Information theory perspective**
   - KL divergence and why cross-entropy makes sense
   - Entropy as uncertainty measure
4. **Practical ML workflow**
   - Train/validation/test splits and hyperparameter tuning

Why MLE Makes Sense – if you know KL-divergences

Intuitive justification:
If the true data distribution is $p^*_{\text{true}}(y \mid \mathbf{x})$ and we approximate it with $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$, then MLE minimizes the **KL divergence**:

$$\text{KL}(p^*_{\text{true}} \| p_{\boldsymbol{\theta}}) = \mathbb{E}_{(\mathbf{x},y)\sim p^*_{\text{true}}} \left[\log \frac{p^*_{\text{true}}(y \mid \mathbf{x})}{p_{\boldsymbol{\theta}}(y \mid \mathbf{x})} \right]$$

Breaking this down:

$$\text{KL}(p^*_{\text{true}} \| p_{\boldsymbol{\theta}}) = \mathbb{E}\left[\log p^*_{\text{true}}(y \mid \mathbf{x})\right] - \mathbb{E}\left[\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x})\right]$$

First term doesn't depend on $\boldsymbol{\theta}$, so minimizing KL is equivalent to:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},y)\sim p^*_{\text{true}}} \left[\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x})\right] \approx \max_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(y_i \mid \mathbf{x}_i)$$

Conclusion: MLE finds the parameters that make our model "closest" to the true distribution (in terms of KL divergence)