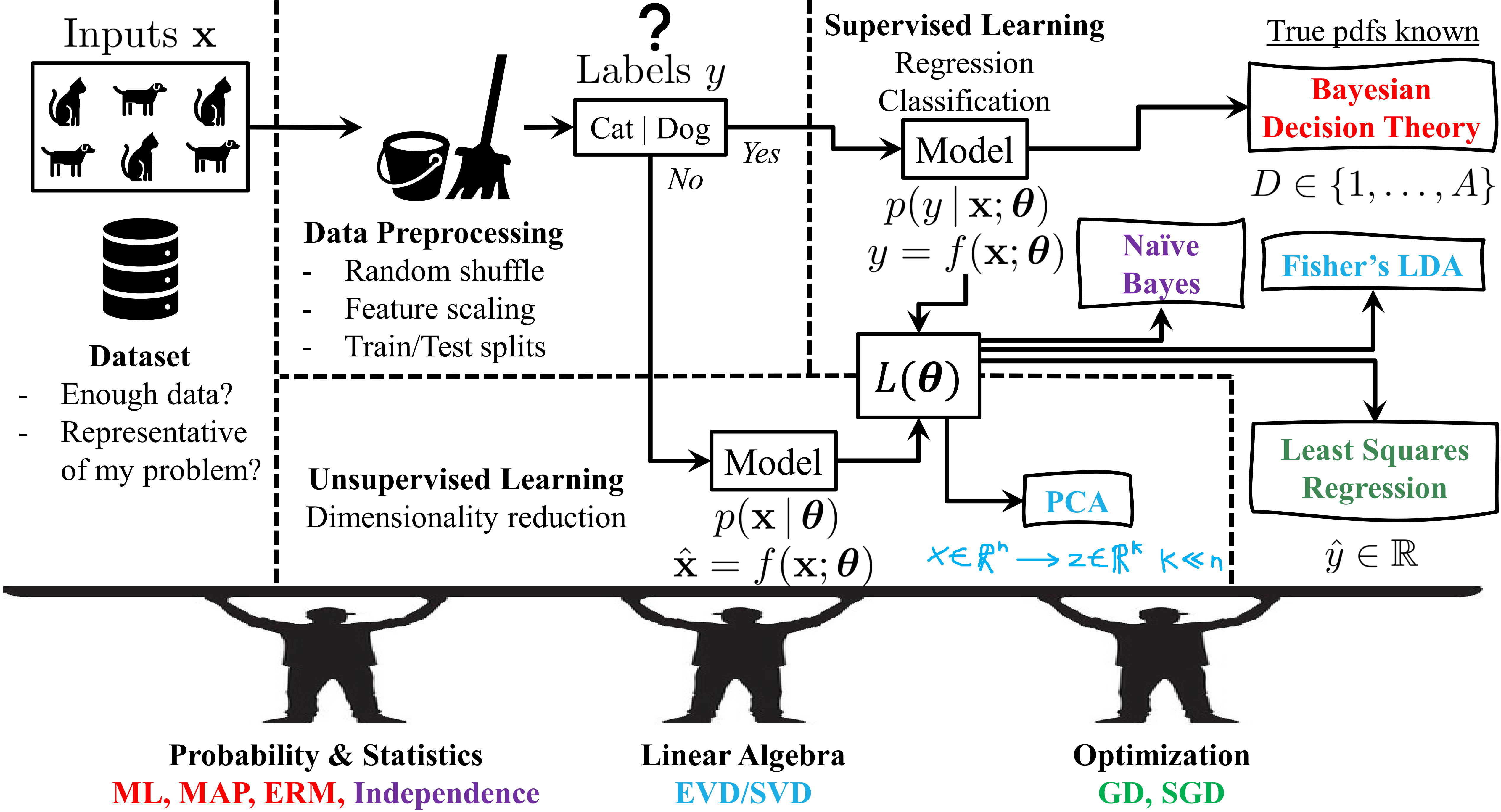
EECE 5644: Bayesian Parameter Estimation

Mark Zolotas

E-mail: m.zolotas@northeastern.edu
Webpage: https://coe.northeastern.edu/people/zolotas-mark/

Tentative Course Outline (Wks. 3-4)

Topics	Dates	Assignments	Additional Reading
~~Naïve Bayes Classifier &~~ ~~*Homework 0 Practice Lab*~~	~~07/18~~	**Homework 2** released on Canvas on 07/22 **Due 08/01**	~~N/A~~
Model Fitting/Training: Bayesian Parameter Estimation	07/19-20		Chpts. 4.1-4.3, 8.7.2-3 Murphy 2022
Logistic Regression	07/21		Chpt. 10 Murphy 2022
Model Selection: Hyperparameter Tuning, k-fold Cross-Validation	07/25	**Homework 3** released on Canvas on 07/29 **Due 08/08**	Chpts. 4.5, 5.2, 5.4.3 Murphy 2022
Regularization, Ridge and Lasso Regression	07/26		Chpts. 4.5, 11.1-11.4 Murphy 2022
Neural Networks: Multilayer Perceptrons & Backpropagation	07/27-28		Chpts. 13.1-13.5 Murphy 2022

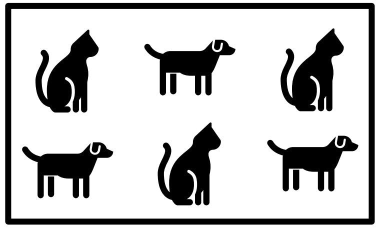


Inputs $\mathbf{x}$
Dataset
- Enough data?
- Representative of my problem?
Data Preprocessing
- Random shuffle
- Feature scaling
- Train/Test splits
?
Labels y
Cat | Dog
Yes
No
Supervised Learning
Regression
Classification
Model
$p(y \mid \mathbf{x}; \boldsymbol{\theta})$
$y = f(\mathbf{x}; \boldsymbol{\theta})$
True pdfs known
Bayesian Decision Theory
$D \in \{1, \ldots, A\}$
Naïve Bayes
Fisher's LDA
$L(\boldsymbol{\theta})$
Least Squares Regression
$\hat{y} \in \mathbb{R}$
Unsupervised Learning
Dimensionality reduction
Model
$p(\mathbf{x} \mid \boldsymbol{\theta})$
$\hat{\mathbf{x}} = f(\mathbf{x}; \boldsymbol{\theta})$
PCA
$x \in \mathbb{R}^n \longrightarrow z \in \mathbb{R}^k \quad k \ll n$
Probability & Statistics
ML, MAP, ERM, Independence
Linear Algebra
EVD/SVD
Optimization
GD, SGD

Bayesian Decisions

Supervised Learning
Regression
Classification

Inputs $\mathbf{x}$



$p(l \,|\, \mathbf{x})$

Model

$$D(\mathbf{x}) = \underset{j \in \{1,\ldots,C\}}{\arg\max}\; p(\mathbf{x} \,|\, l_j)\, p(l_j)$$

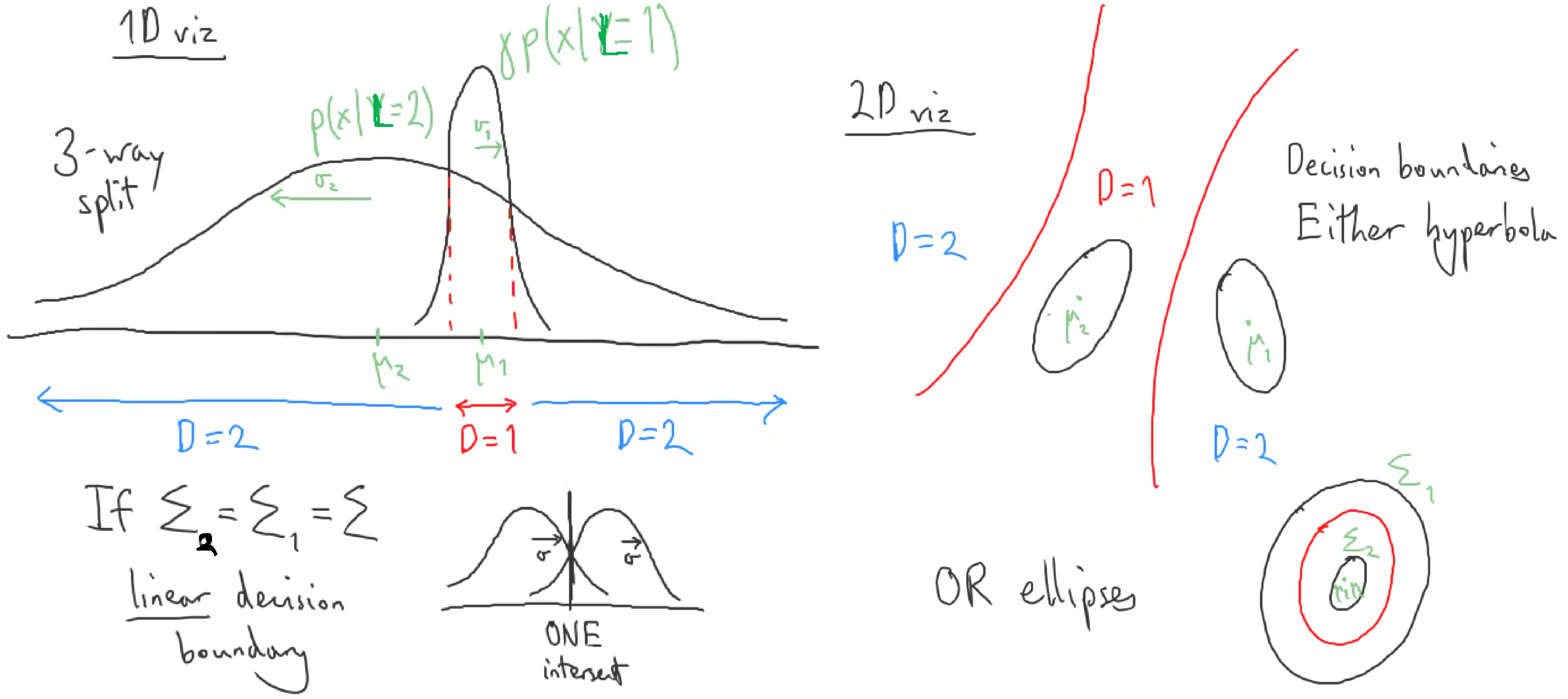
- Make decisions to minimize expected conditional loss or… risk (**ERM**):

$$D(\mathbf{x}) = \underset{i \in \{1,\ldots,A\}}{\arg\min}\; R(d_i \,|\, \mathbf{x}) = \underset{i \in \{1,\ldots,A\}}{\arg\min} \sum_{j=1}^{C} \lambda_{ij}\, p(\mathbf{x} \,|\, l_j)\, p(l_j)$$

- Assume **zero-one** loss → **Max. a Posteriori (MAP)** principle:

$$D(\mathbf{x}) = \underset{j \in \{1,\ldots,C\}}{\arg\max}\; p(l_j \,|\, \mathbf{x})$$

Decision Boundaries Illustration



Why Parameter Estimation?

- Design an **optimal** classifier if we know:
  - Class priors $p(y_j)$
  - Class-conditional likelihoods $p(\mathbf{x} \mid y_j)$

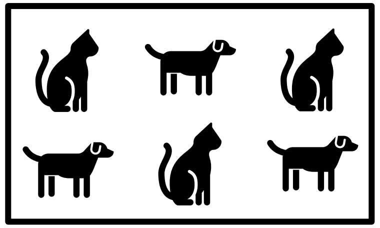
→ Bayesian Decision Theory

True pdfs known

- Rarely know "true" probabilities; but have data to **estimate** them
  - E.g. website visits, heads coin flips, snowfall in Boston
  - **"Estimator"** is a statistic $\hat{\theta}$ that estimates the true population θ
- *Why estimate parameters?* Crux of machine learning and understanding data

Supervised Learning
Regression
Classification

Inputs $\mathbf{x}$



$p(y \mid \mathbf{x}; \boldsymbol{\theta})$

→ Model →

$$\hat{y} = \arg\max_{j \in \{1,\ldots,C\}} p(y_j \mid \mathbf{x}; \boldsymbol{\theta})$$

Parametric Models

- Consider certain probability distributions:
  - $\text{Ber}(p) \rightarrow \theta = p$
  - $\text{Uniform}(a, b) \rightarrow \boldsymbol{\theta} = [a, b]$
  - $N(\mu, \sigma^2) \rightarrow \boldsymbol{\theta} = [\mu, \sigma^2]$
  - …
- These are all **parametric models** or a **parametric family**
- Rather than estimate $\boldsymbol{\theta}$ for arbitrary pdf, assume **known distribution**
- OK… But how do we estimate a Gaussian with μ and σ^2 from our dataset?

Sample Estimators?

- Use sample estimate? E.g. **sample mean** $\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}$

- Maybe fine for class priors $p(y_j)$ but for class-conditional $p(\mathbf{x} \mid y_j)$:
  - Never enough samples to represent your "true" distribution (need very high N)
  - High-dimensional n may cause many problems, **overfitting**, **complexity**, etc.
  - <u>Limited:</u> Not robust enough to fit all parametric distributions, e.g. GMMs

Naive Bayes assumption $X \perp\!\!\!\perp Y \mid Z$ helps

- Need a more general tool to fit our parametric models

Parameter Estimation

- Estimate $\hat{\boldsymbol{\theta}} \in \mathbb{R}^n$ from dataset D (note that D is <u>dataset</u> going forward)
- Known also as **model fitting/training**
- Consider as an **optimization** problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

- Compute estimator $\widehat{\boldsymbol{\theta}}$ according to two familiar loss functions:
  - **Maximum Likelihood (ML)** – Deterministic point estimate $\widehat{\boldsymbol{\theta}}$
  - **Maximum a Posteriori (MAP)** – Random $\widehat{\boldsymbol{\theta}}$ with a prior $p(\widehat{\boldsymbol{\theta}})$
- **Assumptions:** i.i.d. samples and a known parametric form of $p(D \mid \boldsymbol{\theta})$

Maximum Likelihood Estimation (MLE)

- Given i.i.d. samples $\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$ from a dataset, take **log likelihood (LL)**:

$$\mathrm{LL}(\boldsymbol{\theta}) = \log p(\mathcal{D} \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}, y^{(i)} \mid \boldsymbol{\theta})$$

Avoid underflow

- Or if unsupervised, then unconditional: $\mathrm{LL}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta})$

- **Key Idea:** Good values of $\boldsymbol{\theta}$ should assign high probability to D

- Motivates the choice to **MLE criterion**:

$$\hat{\boldsymbol{\theta}}_{\mathrm{MLE}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta})$$

How to compute?

$$\frac{\delta \mathrm{LL}(\theta)}{\delta \theta} = 0$$

Example: Univariate Gaussian – Unknown μ

$$X \sim \mathcal{N}(\mu, \sigma^2) \quad p_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\hat{\mu}_{\mathrm{MLE}} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)} = \overline{x}$$

Sample estimates appear frequently from MLE

Example: Multivariate Bernoulli

$$X \sim \text{Ber}(\theta) \quad p_X(\mathbf{x}) = \prod_{i=1}^{N} \theta^{x^{(i)}} (1-\theta)^{(1-x^{(i)})}$$

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)} = \frac{N_H}{N_H + N_T}$$

N_H ← Heads

N_T ← Tails

Negative Log Likelihood

- Machine learning often looks at **minimization** problems
- Re-define MLE objective as **Negative Log Likelihood (NLL)**:

$$\mathrm{NLL}(\boldsymbol{\theta}) = -\log p(\mathcal{D} \mid \boldsymbol{\theta}) = -\sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta})$$

- Minimization formulation has exact same result:

$$\hat{\boldsymbol{\theta}}_{\mathrm{NLL}} = \arg\min_{\boldsymbol{\theta}} -\sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta})$$

MLE/NLL for Linear Regression

- Probabilistic view of linear regression as a conditional Gaussian:

$$p(y \mid \mathbf{x}; \boldsymbol{\theta}) = \mathcal{N}\big(f_\mu(\mathbf{x}; \boldsymbol{\theta}), f_\sigma(\mathbf{x}; \boldsymbol{\theta})\big)$$

- Two functions of inputs, f_μ and f_σ to predict mean and variance
- Common to assume fixed variance (**homoscedastic regression**) and linear f_μ

$$p(y \mid \mathbf{x}; \boldsymbol{\theta}) = \mathcal{N}\big(\mathbf{w}^\intercal \mathbf{x}, \sigma^2\big)$$

- **NLL** for Gaussian:

$$\mathrm{NLL}(\boldsymbol{\theta}) = \underbrace{\frac{1}{2\sigma^2}}_{\text{const}} \overbrace{\sum_{i=1}^{N} (y^{(i)} - \mathbf{w}^\intercal \mathbf{x}^{(i)})^2}^{\text{RSS}} + \underbrace{\frac{N}{2} \log(2\pi\sigma^2)}_{\text{const}}$$

MLE Algorithm

Very general framework that can be summarized as:

1. Choose parametric model for $p(D \mid \boldsymbol{\theta})$ and define PMF/PDF

$$\hat{\boldsymbol{\theta}}_{\text{NLL}} = \arg\min_{\boldsymbol{\theta}} - \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta})$$

2. Write out log-likelihood function, e.g. NLL and express as an argmax for optimization

3. Use an optimization algorithm, e.g., GD or SGD to calculate argmax and derive $\widehat{\boldsymbol{\theta}}_{MLE}$

MLE Advantages

- ML estimators are **consistent** (converges to true $\boldsymbol{\theta}^*$ at limit) $\hat{\boldsymbol{\theta}} \rightarrow \boldsymbol{\theta}^*$ as $N \rightarrow \infty$
- Good **convergence** properties as training samples N increases
- **Simpler** implementation that other estimators (including Bayesian)
- **Asymptotically optimal** in variance reduction (smallest for large samples)
- Best used in practice when N is large relative to parameter space (careful)

MLE Bias

- May produce **biased** parameter estimates of true parameters
- Bias of estimator $\hat{\theta}$ is how different its expected value is over D from true θ^*

$$\text{bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta^* = 0 \;\therefore\; \text{Unbiased}$$

- MLE example for Gaussian mean and variance:

$$\text{bias}(\hat{\mu}_{\text{MLE}}) = 0 \implies \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} x^{(i)}\right] = \frac{N\mu}{N} = \mu^* \quad \leftarrow \text{Unbiased}$$

$$\text{bias}(\sigma^2_{\text{MLE}}) \neq 0 \implies \mathbb{E}\left[\sigma^2_{\text{MLE}}\right] = \frac{N-1}{N}\sigma^2 \neq \sigma^2 \quad \leftarrow \text{Biased}$$

MLE Overfitting

- **Data sparsity:** Underperforms when there is too little data

- E.g. flip a coin three times and get Heads each time:

$$\hat{\theta}_{\text{MLE}} = \frac{N_H}{N_H + N_T} = \frac{3}{3+0} = 1$$

- Requires a lot of data, else susceptible to poor generalization (**overfitting**)

Coding Break

Bayesian Parameter Estimation

- Main solution to overfitting: **regularization**

More on regularization and overfitting next week

- Results in our **MAP** or **Bayesian parameter** estimate:

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid D) = \arg\max_{\boldsymbol{\theta}} \left[\log p(D \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right]$$

Penalty term for regularization

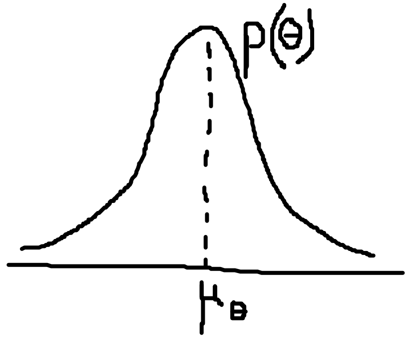
- Add **prior** to MLE estimator, which when **uniform** $\widehat{\boldsymbol{\theta}}_{MAP} = \widehat{\boldsymbol{\theta}}_{MLE}$
- $\widehat{\boldsymbol{\theta}}_{MAP}$ is a **random variable** and has a measure of uncertainty

Treating θ as an RV

- Treats $\boldsymbol{\theta}$ as an **RV** instead of a **point estimate** as in MLE

- Modeling uncertainty about $\boldsymbol{\theta}$ using another distribution; the **prior** $p(\boldsymbol{\theta})$

- Prior $p(\boldsymbol{\theta})$ will even have its own parameters, like a mean



- Any Bayesian model for parameter estimation requires:
  - A **prior** $p(\boldsymbol{\theta})$ encodes beliefs about data BEFORE making observations
  - **Likelihood** $p(D \mid \boldsymbol{\theta})$ follows exactly as in MLE

Bayesian Parameter Posterior

- Remember the MAP rule to infer class posterior probabilities $p(y_j \mid \mathbf{x})$?
- Switch from supervised learning into **unsupervised density estimation** $p(\mathbf{x})$
- **Assumptions:** we have a known prior $p(\boldsymbol{\theta})$ and parametric form $p(D \mid \boldsymbol{\theta})$
- Now **update beliefs** of prior by computing the **posterior** distribution over **continuous-valued** parameters $p(\boldsymbol{\theta} \mid D)$ using Bayes rule:

$$p(\boldsymbol{\theta} \mid D) = \frac{p(D \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(D)} = \frac{p(D \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{\theta})p(D \mid \boldsymbol{\theta})d\boldsymbol{\theta}}$$

Likelihood of data given each set of $\boldsymbol{\theta}$

Prior reflects what we already know about parameters

Rarely compute explicitly as computationally intractable

*Known as **marginal likelihood** of the data*

Coin Setting

- Flip a coin three times and get Heads each time:

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \frac{N_H}{N_H + N_T} = \frac{3}{3+0} = 1$$

- If we use MLE then all future coin tosses will be predicted as heads…

- Example of **overfitting** → Need to specify a prior!

- Picking a prior is <u>challenging</u> for Bayesian parameter estimation:
  - Many techniques but a common one is to choose a **conjugate prior**
  - Allows integrals in posterior to be **analytically tractable**

Conjugate Priors

- Priors $p(\boldsymbol{\theta})$ chosen to **pair** with the likelihood $p(D \mid \boldsymbol{\theta})$ such that the posterior can be computed in *"closed form"* (analytically evaluate integral)
- Name comes from how priors are picked to be **"conjugate"** to the likelihood
- Examples
  - Bernoulli likelihood => **Beta** prior
  - Gaussian likelihood => often **Gaussian** prior or another exponential family

Beta conjugate prior $p(\theta)$ *for Bernoulli RV*

$$\text{Beta}(\theta \mid a, b) \propto \theta^{a-1}(1-\theta)^{b-1}$$

Proportionality to ignore normalization constant

Posterior Distribution for Coin Setting

- Likelihood of D for Bernoulli flip-a-coin scenario:

$$p(D \mid \theta) = \theta^{N_H}(1-\theta)^{N_T}$$

- Given conjugate **beta** prior, can compute posterior: $\mathrm{Beta}(\theta \mid a, b) \propto \theta^{a-1}(1-\theta)^{b-1}$

$$\begin{aligned} p(\theta \mid D) &\propto p(\theta)p(D \mid \theta) \\ &\propto \left[\theta^{a-1}(1-\theta)^{b-1}\right]\left[\theta^{N_H}(1-\theta)^{N_T}\right] \\ &= \theta^{a-1+N_H}(1-\theta)^{b-1+N_T} \end{aligned}$$

- From this posterior distribution over parameters, we can predict new $\hat{\mathbf{x}}$

Posterior Predictive Distribution

- Posterior distribution over future observations given past data:

$$p(\hat{\mathbf{x}} \mid D) = \int p(\boldsymbol{\theta} \mid D) p(\hat{\mathbf{x}} \mid \boldsymbol{\theta}) d\boldsymbol{\theta}$$

- Marginalize out parameters; each parameter setting defines a model

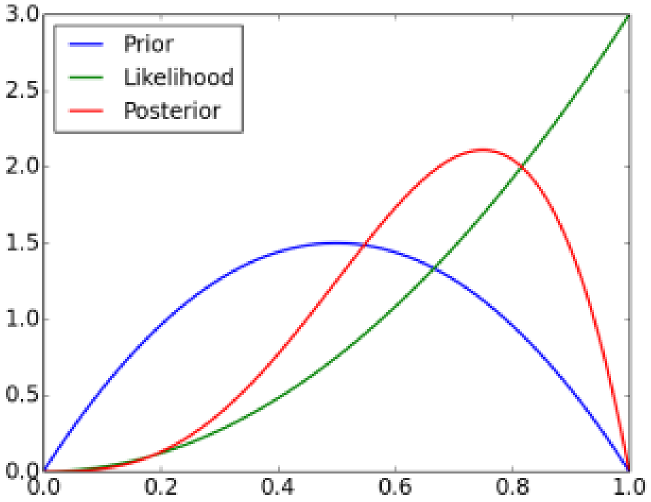
- End up with an **average/expectation** over **all models**

- This is NOT using $\widehat{\boldsymbol{\theta}}_{MLE}$ or $\widehat{\boldsymbol{\theta}}_{MAP}$ but instead all possible $\boldsymbol{\theta}$

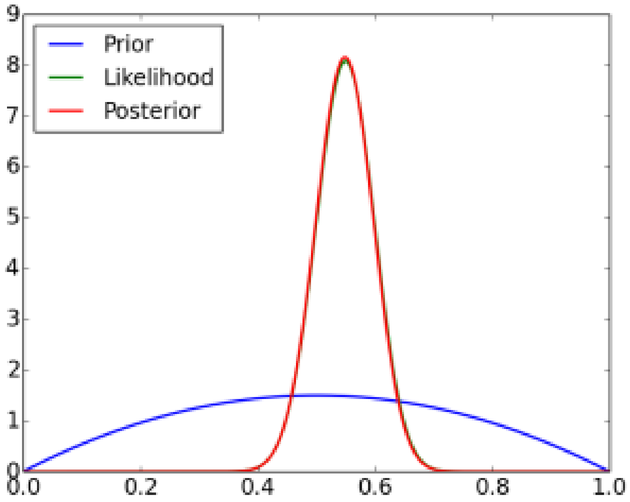
Bayesian Parameter Estimation – Coin Priors

Bayesian inference for the coin flip example:

Small data setting
$N_H = 2,\ N_T = 0$



Large data setting
$N_H = 55,\ N_T = 45$



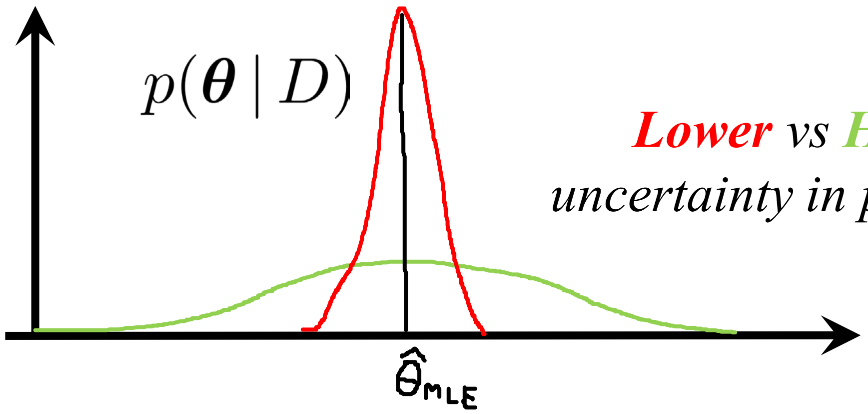
When you have enough observations, the **data overwhelm the prior**.

Sources – UofT CSC 311: Introduction to Machine Learning

Bayesian Parameter Estimation vs MLE (Frequentist)

- **Bayesian** better for **small** N and captures complete parameter representation (most probable estimate AND **uncertainty**) from **single dataset**
- MLE only produces the "best" estimate; needs **repeated experiments**

*If prior **very narrow** can end up similar to MLE*



Lower *vs* ***Higher*** *uncertainty in parameter*

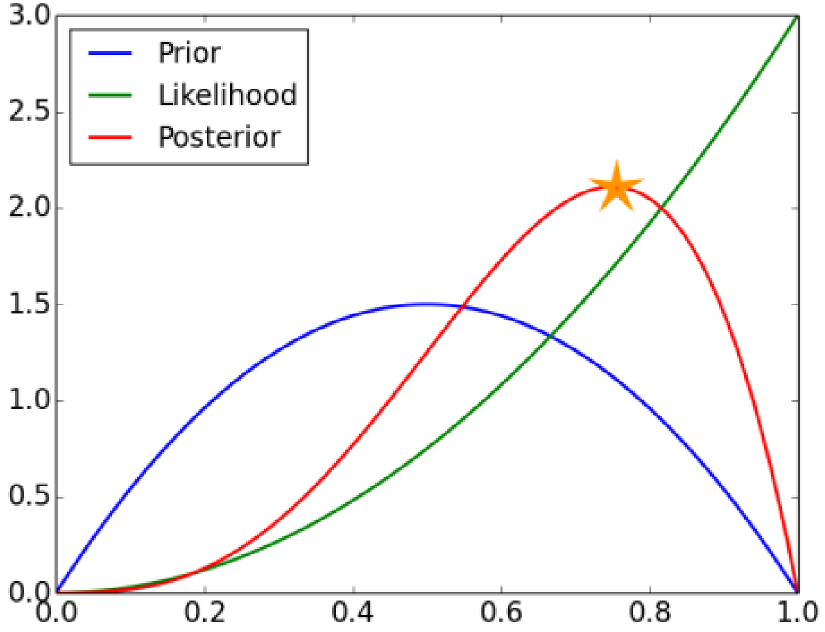
- MLE is optimization-based (e.g. GD) while Bayesian requires integration → much harder in practice, often need approximations, conjugates, etc.

Maximum a Posteriori (MAP) Estimation

- To convert Bayesian parameter estimation into an **optimization** problem, take the most probable parameter estimate (**mode**)

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid D) = \arg\max_{\boldsymbol{\theta}} \left[\log p(D \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right]$$

- Can obtain different loss functions from the posterior distribution
  - Min. MSE => Mean
  - Min. Absolute Error => Median
  - Identical for Gaussian posterior



MAP Estimation for Coin Setting

- Recall log-likelihood of D for Bernoulli flip-a-coin scenario:

$$\mathrm{LL}(\theta) = N_H \log\theta + N_T \log(1-\theta)$$

- Log likelihood plus log of conjugate **beta** prior: $\mathrm{Beta}(\theta \,|\, a, b) \propto \theta^{a-1}(1-\theta)^{b-1}$

$$\mathrm{LL}(\theta) = [N_H \log\theta + N_T \log(1-\theta)] + [(a-1)\log\theta + (b-1)\log(1-\theta)]$$

- Can take derivative of $LL(\theta)$ and equate to 0 to compute MAP estimate:

$$\hat{\theta}_{\mathrm{MAP}} = \frac{N_H + a - 1}{N_H + N_T + a + b - 2} = 0.8 \neq 1$$

- Flip a coin three time with all Heads? $N_H = 3, N_T = 0$ but set $a = b = 2$

Regularization

- We will explore MAP for parameter estimation next week in the context of linear & logistic regression → Usually expressed as **regularization**
- Key idea to add a *complexity penalty term to avoid overfitting*, which is the **prior** from a probability perspective!
- Onto logistic regression!

Concluding Remarks

- Bayesian Parameter Estimation with connection to linear regression!
- Code:

https://github.com/mazrk7/EECE5644_IntroMLPR_LectureCode/blob/main/notebooks/linear_regression/lin_reg_mle.ipynb

- Questions?