Course Logistics and Introduction to Probabilistic Machine Learning

CS772A: Probabilistic Machine Learning

Piyush Rai

Course Logistics

- Course Name: Probabilistic Machine Learning – CS772A
- 2 classes each week
  - Mon/Thur 18:00-19:30
  - Venue: KD-101
- All material (readings etc) will be posted on course webpage (internal access)
  - URL: https://web.cse.iitk.ac.in/users/piyush/courses/pml_autumn22/pml.html
- Q/A and announcements on Piazza. Please sign up
  - URL: https://piazza.com/iitk.ac.in/firstsemester2022/cs772a
  - If need to contact me by email (piyush@cse.iitk.ac.in), prefix subject line with "CS772"
- Auditors are welcome
  - However, can't appear for quizzes/exams. Also, won't be able to grade your homeworks

Workload and Grading Policy

- 4 homeworks (theory + programming): 40%
  - Must be typeset in LaTeX. To be uploaded on Gradescope (login details will be shared)
- 2 quizzes: 10% (tentative dates: Aug 27, Oct 22)
- Mid-sem exam: 20% (date as per DOAA schedule)
- End-sem exam: 30% (date as per DOAA schedule)

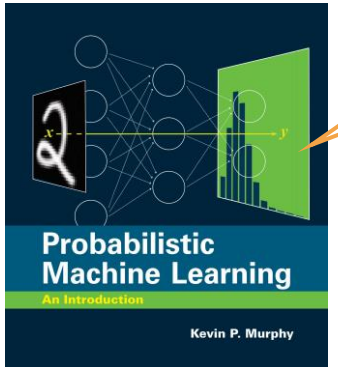
Tentative dates for release of homeworks: Aug 11, Aug 29, Sept 26, Oct 20

Each homework due roughly in 2 weeks (excluding holidays, exam weeks)

- (Optional) Research project (to be done in groups of 2): 20%
  - If opted, will be exempted from appearing in the mid-sem exam
  - We may suggest some topics but the project has to be driven by you
  - We will not be able to provide compute resources
  - Advisable only if you have strong prior experience of working on ML research projects

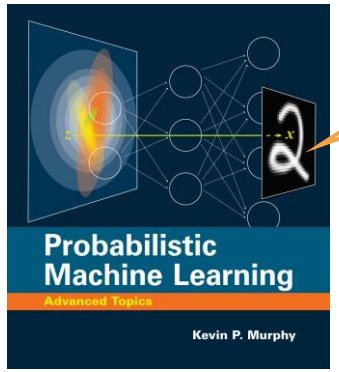
Textbooks and Readings

- Textbook: No official textbook required
- Some books that you may use as reference (freely available PDF)
  - Kevin P. Murphy, Probabilistic Machine Learning: An Introduction (PML-1), The MIT Press, 2022.
  - Kevin P. Murphy, Probabilistic Machine Learning: Advanced Topics (PML-2), The MIT Press, 2022.
  - Christopher M. Bishop, Pattern Recognition and Machine Learning (PRML), Springer, 2007.



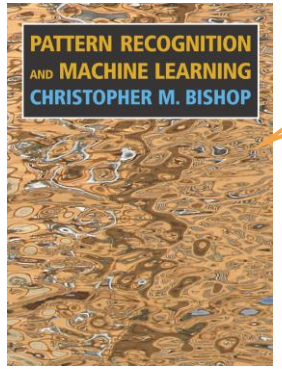
855 pages

Python code



1347 pages

Python code



758 pages

Python code

All these books have accompanying Python code. You are encouraged to explore (we will also go through some)

- Follow the suggested readings for each lecture (may also include some portions from these books), rather than trying to read these books in a linear fashion

Course Policies

- Policy on homeworks
  - Homework solutions must be in your own words
  - Must cite sources you have referred to or used in preparing your HW solutions
  - No requests for deadline extension will entertained. Plan ahead of time
  - Late submissions allowed up to 72 hours with 10% penalty per 24 hour delay
  - Every student entitled for ONE late homework submission without penalty (use it wisely)

- Policy on collaboration/cheating
  - Punishable as per institute's/department's rules
  - Plagiarism from other sources will also lead to strict punishment
  - Both copying as well as helping someone copy will be equally punishable

Course Team

TA

Abhinav Joshi
ajoshi@cse.iitk.ac.in

TA

Abhishek Jaiswal
abhijais@cse.iitk.ac.in

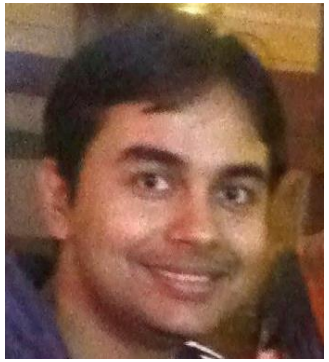
TA

Avideep Mukherjee
avideep@cse.iitk.ac.in

TA

Soumya Banerjee
soumyab@cse.iitk.ac.in

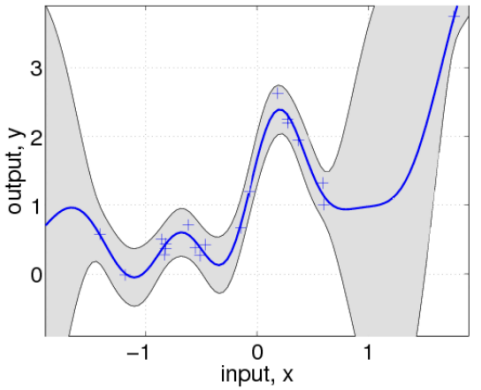
Instructor



Piyush Rai
piyush@cse.iitk.ac.in

Course Goals

- Introduce you to the foundations of probabilistic machine learning (PML)
- Focus will be on learning core, general principles of PML
  - How to set up a probabilistic model for a given ML problem
  - How to quantify uncertainty in parameter estimates and predictions
  - Estimation/inference algorithms to learn various unknowns in a model
  - How to think about trade-offs (computational efficiency vs accuracy)
- Will also look at the above using specific examples of ML models
- Throughout the course, focus will also be on contemporary/latest advances
  - PML is a fast-moving field, especially in the era of modern machine/deep learning



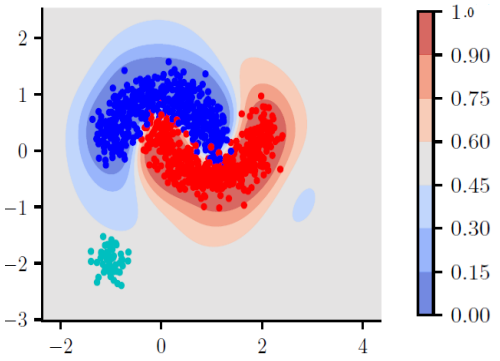


Image source: Probabilistic Machine Learning – Advanced Topics (Murphy, 2022)

Why Probabilistic ML?

Probabilistic ML because… Uncertainty Matters!

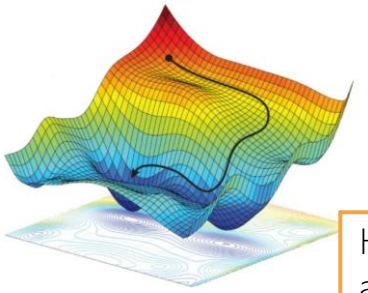
- ML models ingest data, give us predictions, trends in the data, etc
- The standard approach: Minimize a loss func. to find optimal parameters. Use them to predict

Single answer! No estimate of uncertainty about true θ

May be unreliable and can overfit especially with limited training data

Training data

$$\hat{\theta} = \arg\min_{\theta} \ell(\mathcal{D}, \theta)$$

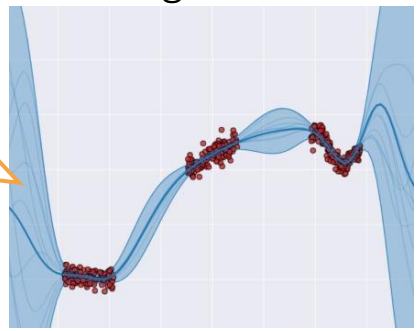


Healthcare applications, autonomous driving, etc.

- In many applications, we also care about the parameter/predictive uncertainty

Regression

Desirable: An approach that reports large predictive uncertainty (variance) in regions where there is little/no training data



Binary classification

Desirable: An approach that reports close to uniform class probability (0.5 in this case) for inputs faraway from training data or for out-of-distribution inputs

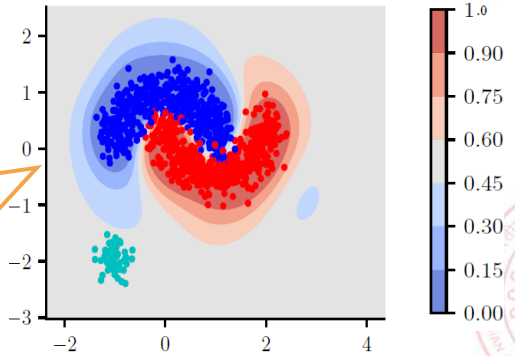


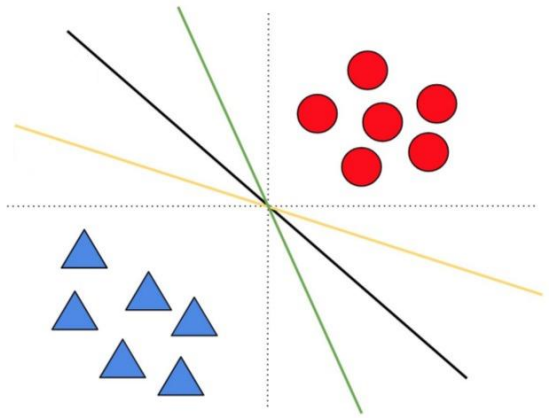
Image source: Probabilistic Machine Learning – Advanced Topics (Murphy, 2022)

Types of Uncertainty: Model Uncertainty

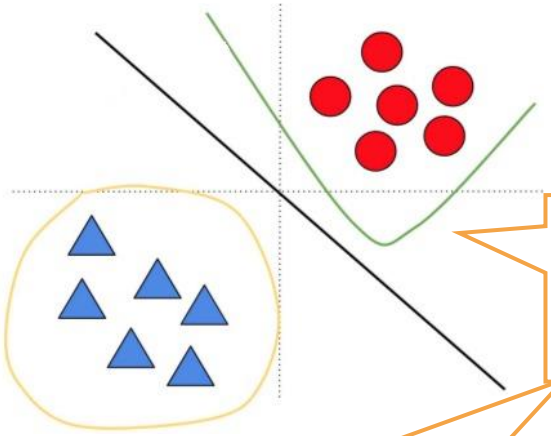
Means uncertainty in model weights

- Model uncertainty is due to not having enough training data

Same model class (linear models) but uncertainty about the weights



Uncertainty not just about the weights but also the model class



3 different model classes considered here (with linear, polynomial, circular decision boundaries)

Each model class itself will have uncertainty (like left fig) since there isn't enough training data

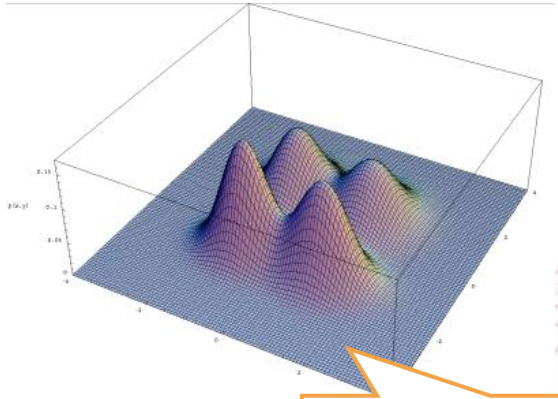
Also known as the "posterior distribution"

Distribution of model parameters conditioned on the training data

Hard to compute in general. Will study several methods to do it

A probabilistic way to express model uncertainty

$$p(\theta|\mathcal{D})$$



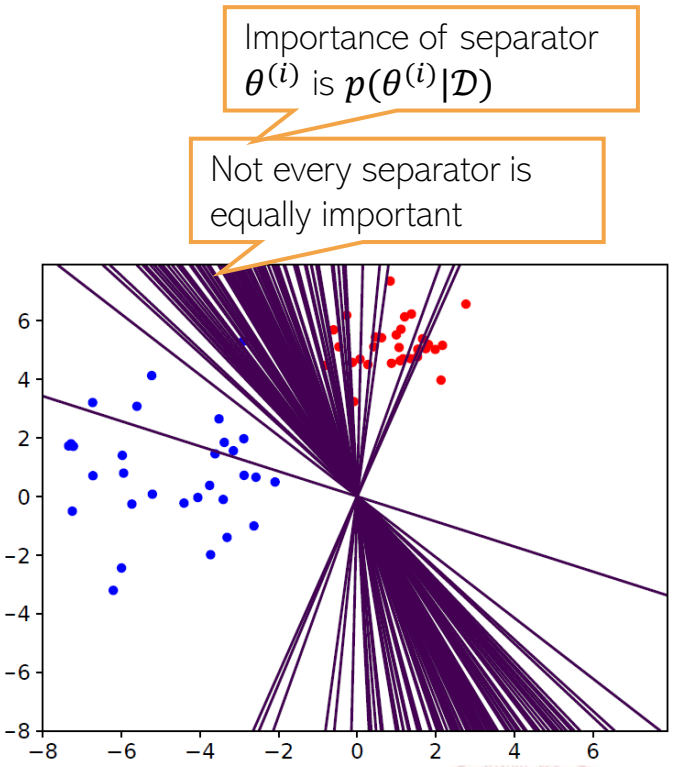
An example posterior of a two-dim parameter vector θ

- Also called epistemic uncertainty. Usually reducible
  - Vanishes with "sufficient" training data

Image credit: Balaji L, Dustin T, Jasper N. (NeurIPS 2020 tutorial)

Model Uncertainty via Posterior: An Illustration

- Consider a linear classification model for 2-dim inputs
- Classifier weight will be a 2-dim vector $\theta = [\theta_1, \theta_2]$
- Its posterior will be some 2-dim distribution $p(\theta|\mathcal{D})$
- Sampling from this distribution will generate 2-dim vectors
- Each vector will correspond to a linear separator (right fig)
- Thus posterior in this case is equivalent to a "collection" or "ensemble" of weights, each representing a different linear separator (will discuss later how to use such a "collection" when making predictions for a test input)



Types of Uncertainty: Data Uncertainty

- Data uncertainty can be due to various reasons, e.g.,
  - Intrinsic hardness of labeling, class overlap
  - Labeling errors/disagreements (for difficult training inputs)
  - Noisy or missing features

A probabilistic way to express data uncertainty

$$p(y|\theta, \boldsymbol{x})$$

Usually specified by the distribution of data being modeled conditioned on model parameters and other inputs

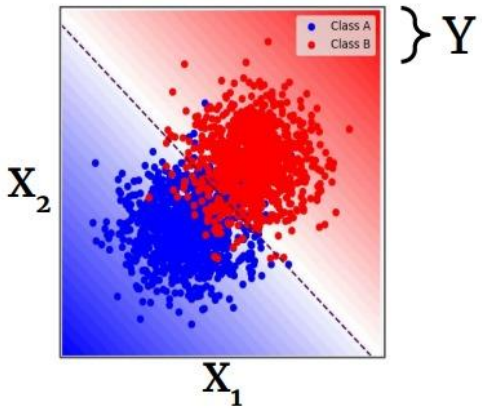


Image credit: Eric Nalisnick

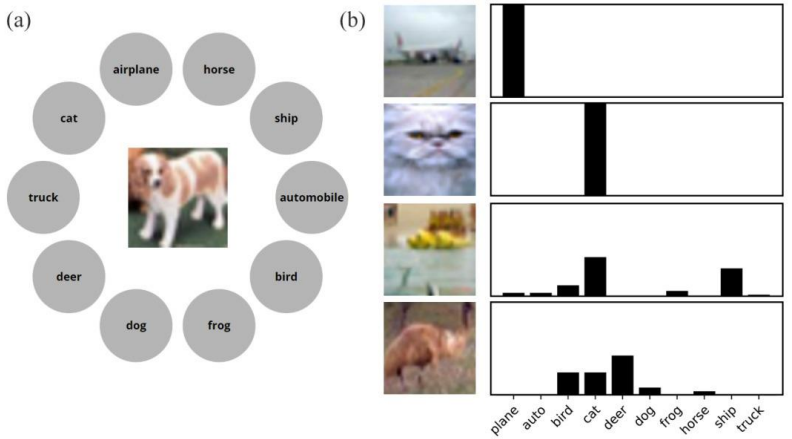


Image source: "Improving machine classification using human uncertainty measurements" (Battleday et al, 2021)

- Also called aleatoric uncertainty. Usually irreducible
  - Won't vanish even with infinite training data
  - Note: Can sometimes vanish by adding more features (figure on the right) or switching to a more complex model

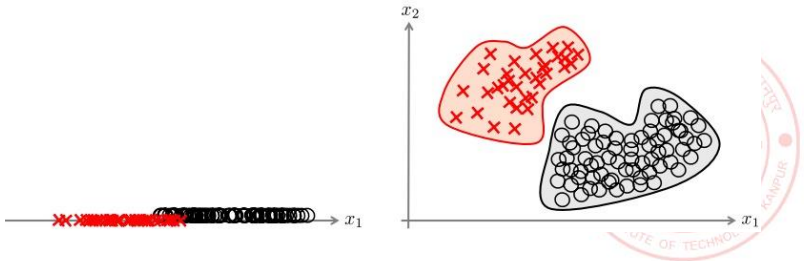


Image source: "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods" (H&W 2021)

A Key Principle of PML: **Marginalization**

- Consider training data $\mathcal{D} = (\mathbf{X}, \mathbf{y})$
- Consider a model m (e.g., a deep neural net) of the form $p(y|x, \theta, m)$

e.g., a deep net with softmax outputs

Note: m is just a model identifier; can ignore when writing

- Let $\hat{\theta}$ denote the optimal weights of model m by minimizing a loss fn. on $\mathcal{D}$
- Standard prediction for a new test input x_* : $p(y_*|x_*, \hat{\theta}, m)$

Uncertainty about θ ignored

For multi-class classification, this quantity will be a vector of predicted class probabilities

- A more robust prediction can be obtained via marginalization

Posterior distribution of the weights

Denotes data / aleatoric uncertainty

Denotes model weight / epistemic uncertainty

$$p(y_*|x_*, \mathcal{D}, m) = \int p(y_*|x_*, \theta, m)p(\theta|\mathcal{D}, m)d\theta$$

For multi-class classification, this too will be a vector of predicted class probabilities but its computation incorporates the uncertainty in θ

Prediction via marginalizing the predictions by each value of θ over its posterior distribution

Predictions by different θ's not weighted equally but based on how likely each θ is given data $\mathcal{D}$,, i.e., $p(\theta|\mathcal{D}, m)$

Will derive the expression shortly – just a simple application of product and chain rules of probability

More on Marginalization

- Marginalization (integral) is usually intractable and needs to be approximated

$$p(y_*|\boldsymbol{x}_*, \mathcal{D}, m) = \int p(y_*|\boldsymbol{x}_*, \theta, m)p(\theta|\mathcal{D}, m)d\theta$$

$$\approx \frac{1}{S}\sum_{i=1}^{S} p\left(y_*|\boldsymbol{x}_*, \theta^{(i)}, m\right)$$

Each $\theta^{(i)}$ is drawn i.i.d. from the distribution $p(\theta|\mathcal{D}, m)$

Above integral replaced by a "Monte-Carlo Averaging"

Will look at these ideas in more depth later

- Marginalization can be done even over several choices of models

Marginalization over all weights of a single model m

$$p(y_*|\boldsymbol{x}_*, \mathcal{D}, m) = \int p(y_*|\boldsymbol{x}_*, \theta, m)p(\theta|\mathcal{D}, m)d\theta$$

Marginalization over all finite choices $m = 1,2,\ldots,M$ of the model

$$p(y_*|\boldsymbol{x}_*, \mathcal{D}) = \sum_{m=1}^{M} p(y_*|\boldsymbol{x}_*, \mathcal{D}, m)p(m|\mathcal{D})$$

For example, deep nets with different architectures

Like a double averaging (over all model choices, and over all weights of each model choice)

Haven't yet told you how to compute this quantity but will see shortly

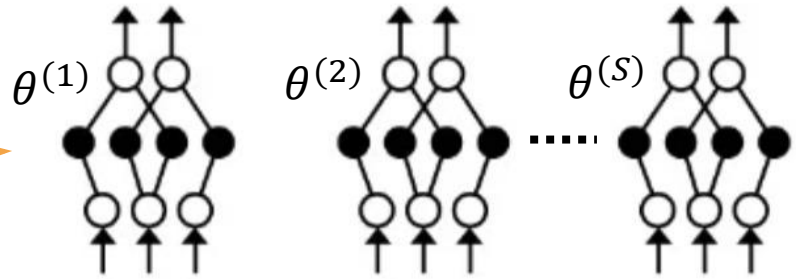
Marginalization is like Ensemble-based Averaging

- Recall the approximation of the marginalization

$$p(y_*|\boldsymbol{x}_*, \mathcal{D}, m) \approx \frac{1}{S}\sum_{i=1}^{S} p\left(y_*|\boldsymbol{x}_*, \theta^{(i)}, m\right)$$

- Akin to computing averaged prediction using ensemble of weights $\left\{\theta^{(i)}\right\}_{i=1}^{S}$

One way to get such an ensemble is to train the same model with S different initializations



Tip: If you can't design/use a full-fledged, rigorous probabilistic model, consider using an ensemble* instead

May not be as good as doing proper marginalization over the posterior but usually better than using a single estimate of the weights

We will also study ensembles later during this course

- Ensembling is a well-known classical technique for strong predictive performance

*Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles (Lakshminarayanan et al, 2017)

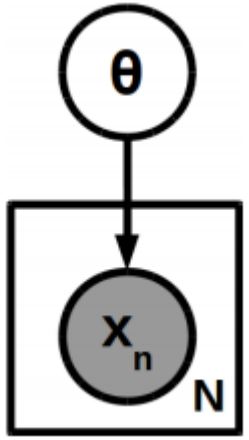
Probabilistic ML: The Basic Set-up

Modeling Data Probabilistically: A Simplistic View

- Assume data $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ generated from a prob. model with params $\boldsymbol{\theta}$

No outputs/labels in this problem; just modeling the inputs (an unsupervised learning task)

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N \sim p(\boldsymbol{x}|\theta)$$

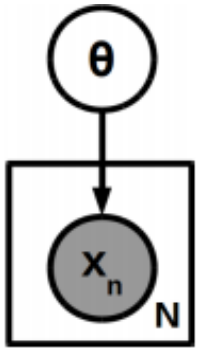


A plate diagram of this simplistic model

- Note: Shaded nodes = observed; unshaded nodes = unknown/unobserved
- Goal: To estimate the unknowns ($\boldsymbol{\theta}$ in this case), given the observed data $\mathbf{X}$
  - Many ways to do this (point estimate or the posterior distribution of $\boldsymbol{\theta}$)
- Can use the parameter estimates to make predictions, e.g.,
  - Probability density of a new input $\boldsymbol{x}_*$ under this model

Basic Ingredients in Probabilistic ML

- Specification of prob. models requires two key ingredients: Likelihood and prior



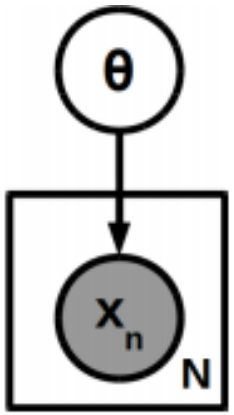
- Likelihood $p(\boldsymbol{x}|\theta)$ or the "observation model" specifies how data is generated
  - Measures data fit (or "loss") w.r.t. the given θ (will see the reason formally later)
- Prior distribution $p(\theta)$ specifies how likely different parameter values are *a priori*
  - Also corresponds to imposing a "regularizer" over θ (will see the reason formally later)
- Domain knowledge can help in the specification of the likelihood and the prior
  - A key benefit of probabilistic modeling

Parameter Estimation in Probabilistic Models

- A simple way: Find θ for which the observed data is most likely or most probable

In this course, the use of the word "inference" would mean Bayesian inference, i.e., computing the (usually an approximate) posterior distribution of the model parameters.

At some other places, especially in deep learning community, the word "inference" usually refers to making prediction on the test inputs



$$\hat{\theta} = \arg\max_{\theta} \log p(\mathbf{X}|\theta)$$

Log likelihood

This "point estimate", called maximum likelihood estimate (MLE), however, does not provide us any information about uncertainty in θ

Just like loss function minimization approach

- More desirable: Estimate the full posterior distribution over θ to get the uncertainty

Fully Bayesian inference. In general, an intractable problem (mainly because computing the marginal $p(\mathbf{X})$ is hard), except for some simple cases (will study how to solve such problems)

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} \propto \text{Likelihood} \times \text{Prior}$$

Can also find a single best estimate of θ by finding the mode of the posterior, i.e., $\hat{\theta} = \arg\max_{\theta} \log p(\theta|\mathbf{X})$, called the maximum-a-posteriori (MAP) estimate, but we won't get uncertainty estimate

- To make predictions, can use full posterior rather than a point estimate (MLE/MAP) θ
  - This is typically referred to as posterior averaging

Posterior Averaging

- Can now use the posterior over params to compute "averaged prediction", e.g.,

Future (test) data

Past (training) data

$$p(\boldsymbol{x}_*|\mathbf{X}) = \int p(\boldsymbol{x}_*|\theta)p(\theta|\mathbf{X})d\theta$$

Posterior predictive distribution (PPD) obtained by doing an importance-weighted averaging over the posterior

Plug-in predictive distribution

"Plug-in" because we plugged-in a single value of θ to compute this

Tells us how important this value of θ is

$$\approx \frac{1}{S}\sum_{i=1}^{S} p(\boldsymbol{x}_*|\theta^{(i)})$$

An approximation of the PPD

Note: The PPD will take different forms depending on the problem/model, e.g., for a discriminative supervised learning model, PPD will be of the form $p(\boldsymbol{y}|\boldsymbol{x}, \mathcal{D})$ where $\mathcal{D}$ is the labeled training data

Proof of the PPD expression

$$p(\boldsymbol{x}_*|\mathbf{X}) = \int p(\boldsymbol{x}_*, \theta|\mathbf{X})\, d\theta$$

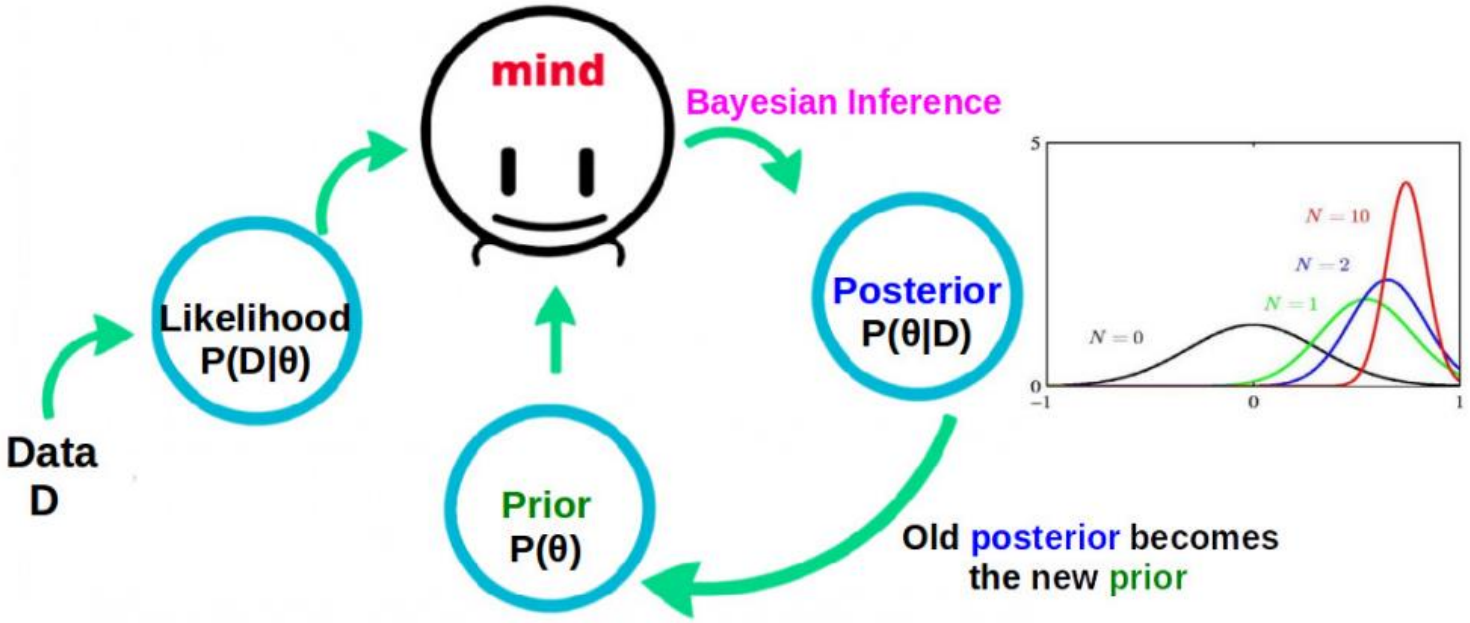
$$= \int p(\boldsymbol{x}_*|\theta, \mathbf{X})p(\theta|\mathbf{X})\, d\theta$$

Assuming observations are i.i.d. given θ

$$= \int p(\boldsymbol{x}_*|\theta)p(\theta|\mathbf{X})\, d\theta$$

Bayesian Inference

- Bayesian inference can be seen in a sequential fashion

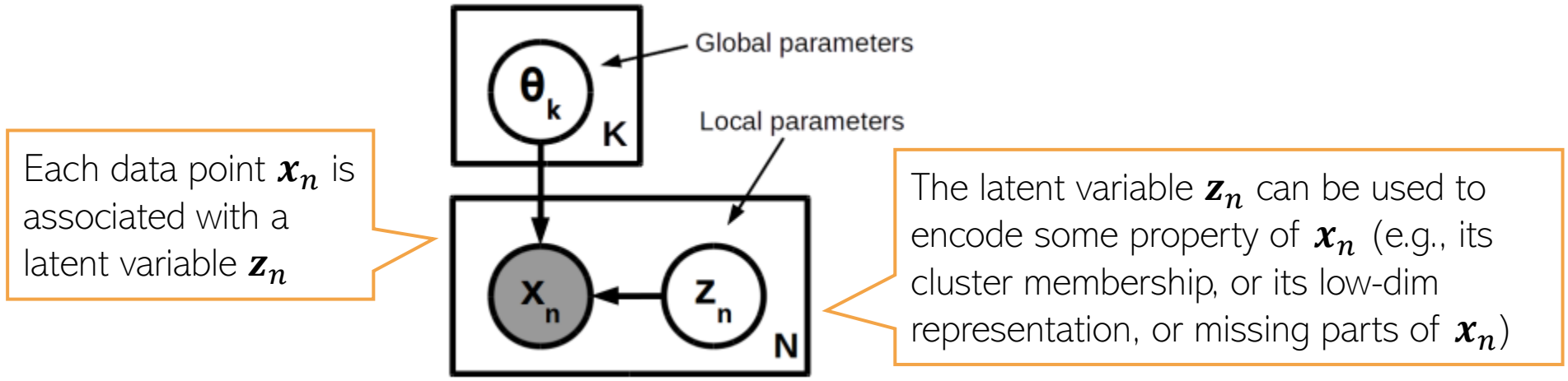


- Our belief about θ keeps getting updated as we see more and more data
  - Posterior keeps getting updates as more and more data is observed
  - Note: Updates may not be straightforward and approximations may be needed

Other Examples of Probabilistic Models of Data

Modeling Data Probabilistically: With Latent Vars

- Can endow generative models of data with latent variables. For example:

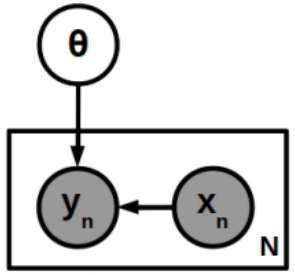


- Such models are used in many problems, especially unsupervised learning: Gaussian mixture model, probabilistic PCA, topic models, deep generative models, etc.

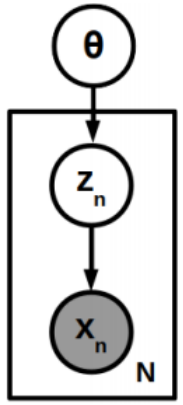
- We will look at several of these in this course and way to learn such models

Modeling Data Probabilistically: Other Examples

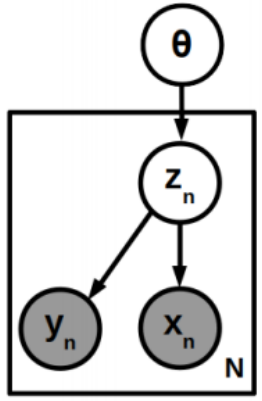
- Can construct various other types of models of data for different problems



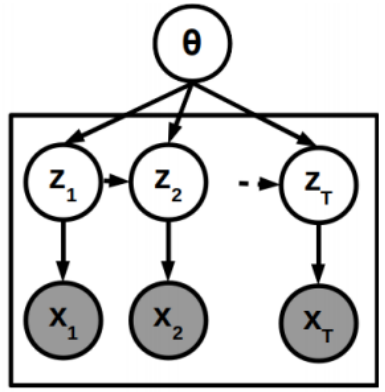
A simple supervised learning model



A latent variable model for unsupervised learning

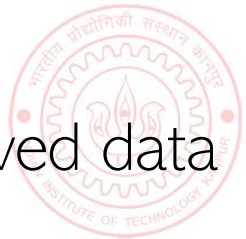


A latent variable model for supervised learning



A latent variable model for sequential data

- Any node (even if observed) we are uncertain about is modeled by a prob. distribution
  - These nodes become the random variables of the model
- The full model is specified via a joint prob. distribution over all random variables
- The goal is to infer the distribution of unknowns of the model, given the observed data

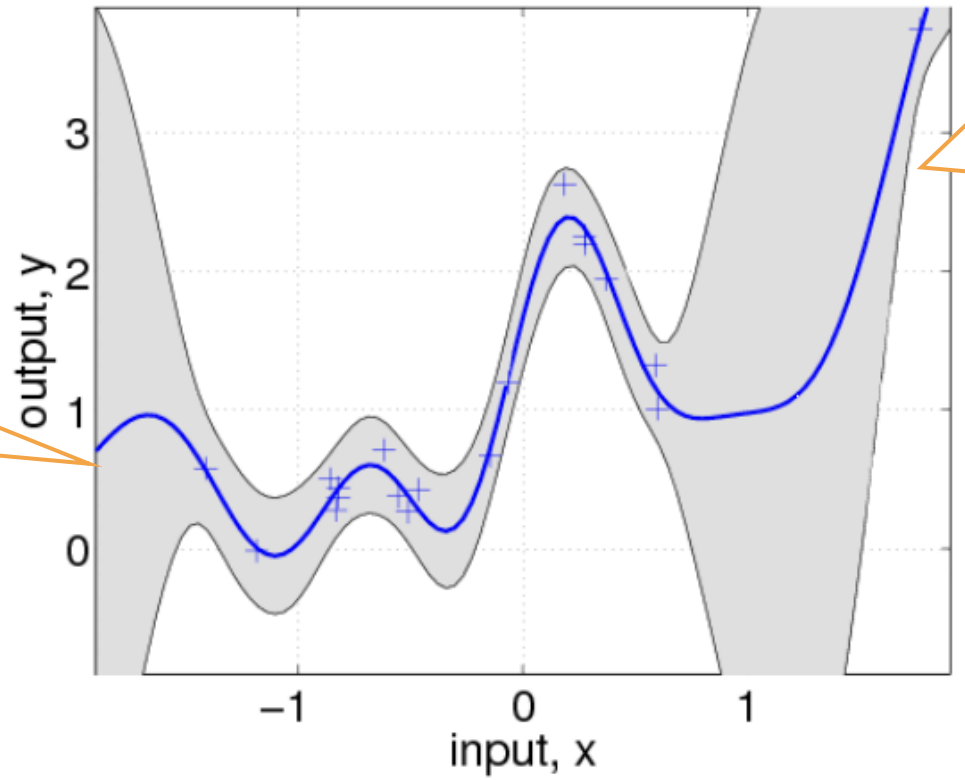


Use probabilistic ML also because..

Helps in Sequential Decision-Making Problems

- Sequential decision-making: Information about uncertainty can "guide" us, e.g.,

Blue curve is the mean of the function, shaded region denotes the predictive uncertainty



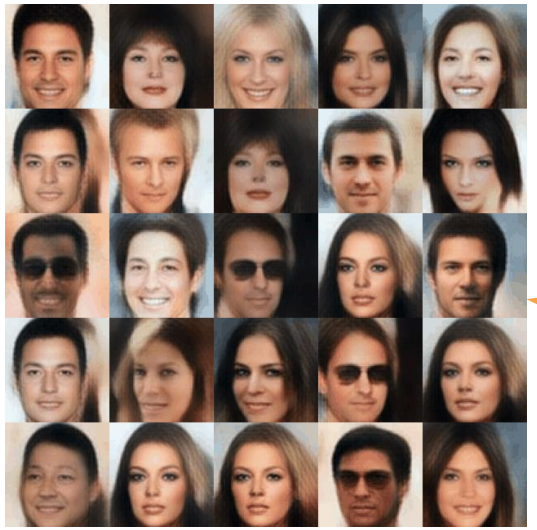
Given our current estimate of the regression function, which training input(s) should we add next to improve its estimate the most?

Uncertainty can help here: Acquire training inputs from regions where the function is most uncertain about its current predictions

- Applications in active learning, reinforcement learning, Bayesian optimization, etc

Can Learn Data Distribution and Generate Data

- Often wish to learn the underlying probability distribution $p(\boldsymbol{x})$ of the data
- Useful for many tasks, e.g.,
  - Outlier/novelty detection: Outliers will have low probability under $p(\boldsymbol{x})$
  - Can sample from this distribution to generate new "artificial" but realistic-looking data

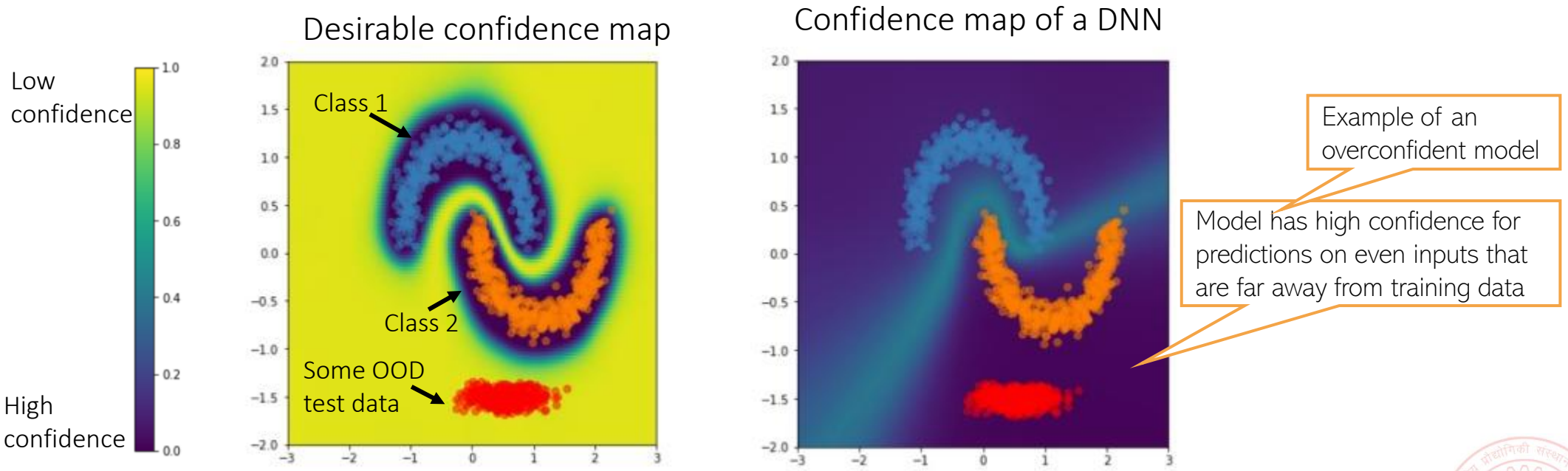


Several models, such as generative adversarial networks (GAN), variational auto-encoders (VAE), denoising diffusion models, etc can do this

Pic credit: https://medium.com/analytics-vidhya/an-introduction-to-generative-deep-learning-792e93d1c6d4

Can Better Handle OOD Data

- Many modern deep neural networks (DNN) tend to be overconfident
- Especially true if test data is "out-of-distribution (OOD)"



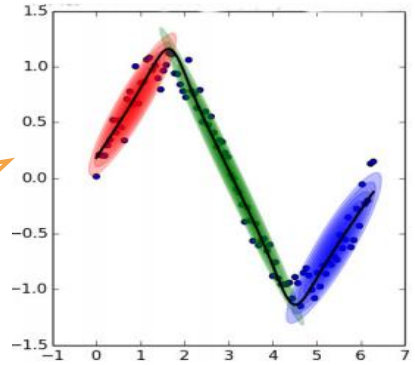
- PML models are robust and give better uncertainty estimates to flag OOD data

Image source: "Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness" (Liu et al, 2020)

Can Construct Complex Models in Modular Ways

- Can combine multiple simple probabilistic models to learn complex patterns

A combination of a mixture model for clustering and a probabilistic linear regression model: Result is a probabilistic nonlinear regression model



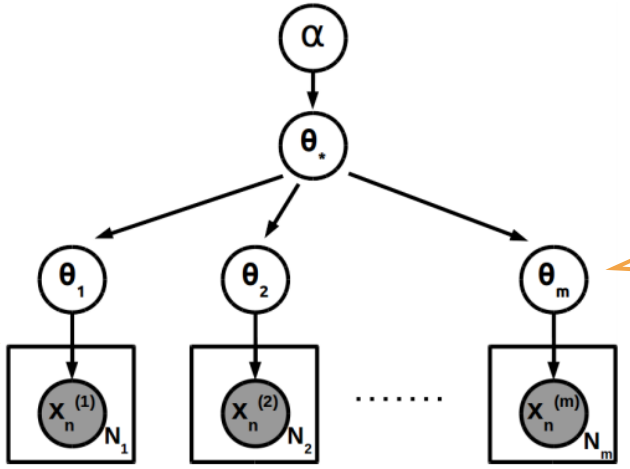
Can design a latent variable model to do this

Essentially a "mixture of experts" model

- Can design models that can jointly learn from multiple datasets and share information across multiple datasets using shared parameters with a prior distribution

Example: Estimating the means of $\boldsymbol{m}$ datasets, assuming the means are somewhat related. Can do this jointly rather than estimating independently

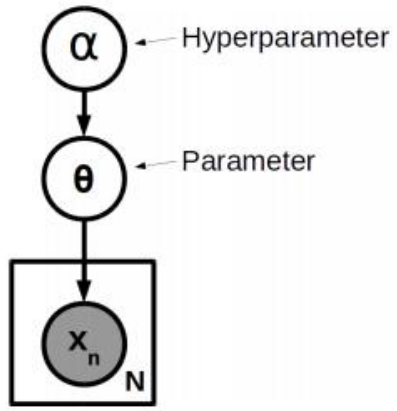
Easy to do it using a probabilistic approach with shared parameters (will see details later)



An example of transfer learning or multitask learning using a probabilistic approach

Hyperparameter Estimation

- ML models invariably have hyperparams, e.g., regularization/kernel h.p. in a linear/kernel regression, hyperparameters of a deep neural network, etc.
- Can specify the hyperparams as additional unknown of the probabilistic model



The approach of using marginal likelihood for doing such thing has some issues; other quantities can be used such as "conditional" marginal likelihood* (more on this later)

A way to find the point estimate of the hyperparameters by maximizing the marginal likelihood of data (more on this later)

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmax}} \log p(\mathbf{X}|\alpha)$$

$$= \underset{\alpha}{\operatorname{argmax}} \log \int p(\mathbf{X}|\theta)p(\theta|\alpha)\theta$$

- Can now estimate them, e.g., using a point estimate or a posterior distribution
  - To find point estimate of hyperparameters, we can write the probability of data as a function of hyperparameters and maximize this quantity w.r.t. the hyperparameters (details later)
  - Posterior can also be estimated if we specify a prior on the hyperparameters as well (details later)

*Bayesian Model Selection, the Marginal Likelihood, and Generalization (Lotfi et al, 2022)

Model Comparison

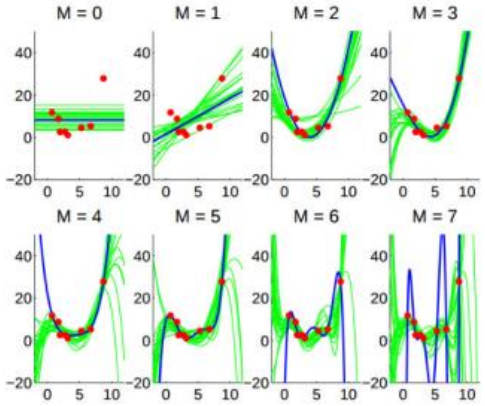
- Suppose we have a number of models $m = 1,2,\ldots,M$ to choose from
- The standard way to choose the best model is cross-validation
- Can also compute the posterior probability of each candidate model, using Bayes rule

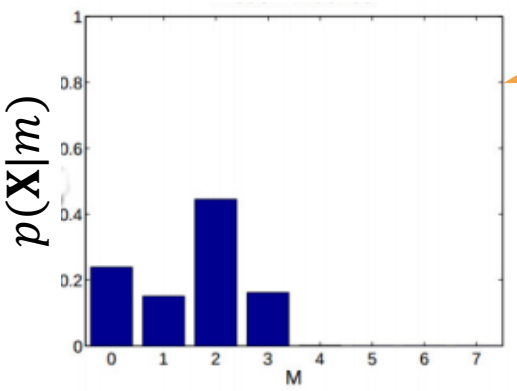
May not be easy to do exactly but can compute it approximately

$$p(m|\mathbf{X}) = \frac{p(m)p(\mathbf{X}|m)}{p(\mathbf{X})}$$

Marginal likelihood of model m

- If all models are equally likely a priori ($p(m)$ is uniform) then the best model can be selected as the one with largest marginal likelihood





This doesn't require a separate validation set unlike cross-validation

Therefore also useful for doing model selection/comparison for unsupervised learning problems

Non-probabilistic ML Methods?

- Some non-probabilistic ML methods can give probabilistic answers via heuristics
- Doesn't mean these methods are not useful/used but they don't follow the PML paradigm, so we won't study them in this course
- Some examples which you may have seen
  - Converting distances from hyperplane (in hyperplane classifiers) to compute class probabilities
  - Using class-frequencies in nearest neighbors to compute class probabilities
  - Using class-frequencies at leaves of a Decision Tree to compute class probabilities
  - Soft k-means clustering to compute probabilistic cluster memberships

Or methods like Platt Scaling used to get class probabilities for SVMs

Tentative Outline

- Basics of probabilistic modeling and inference
  - Common probability distributions
  - Basic point estimation (MLE and MAP)
- Bayesian inference (simple and not-so-simple cases)
- Probabilistic models for regression, classification, clustering, dimensionality reduction
- Gaussian Processes (probabilistic modeling meets kernels)
- Latent Variable Models (for i.i.d., sequential, and relational data)
- Approximate Bayesian inference (EM, variational inference, sampling, etc)
- Bayesian Deep Learning
- Misc topics, e.g., deep generative models, black-box inference, sequential decision-making, etc