Course Logistics and Introduction to Probabilistic Machine Learning

CS772A: Probabilistic Machine Learning

Piyush Rai

Course Logistics

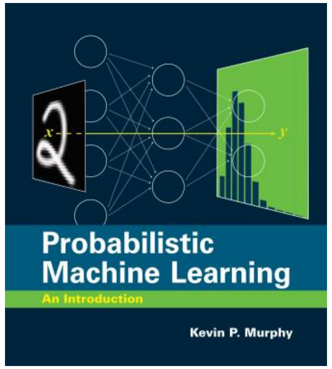
- Course Name: Probabilistic Machine Learning – **CS772A**
- 2 classes each week
  - Mon/Thur 18:00-19:15
  - Venue: RM-101
- Attendance policy: None but biometric attendance will be taken
- All material (readings etc) will be posted on course webpage (internal access)
  - URL: https://web.cse.iitk.ac.in/users/piyush/courses/pml_spring25/pml.html
- Q/A and announcements on Piazza. Please sign up
  - URL: https://piazza.com/iitk.ac.in/secondsemester2025/cs772
  - If need to contact me by email (piyush@cse.iitk.ac.in), prefix subject line with "CS772"
- Unofficial auditors are welcome

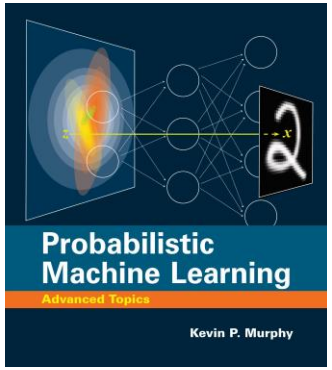
Workload and Grading Policy

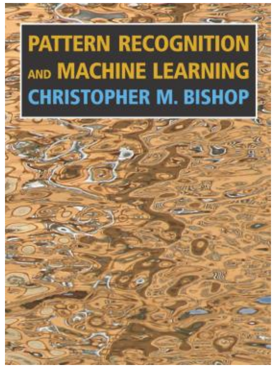
- 3 quizzes: 30%
  - In class, closed-book
- Mid-sem exam: 20% (date as per DOAA schedule). Closed book
- End-sem exam: 30% (date as per DOAA schedule). Closed book
- Research project (to be done in groups of 4-5): 20%
  - Some topics will be suggested (research papers)
  - You can propose your own topic (but must be related to probabilistic ML)
  - More details will be shared soon
- Proration: If you miss any quiz/mid-sem, we can prorate it using end-sem marks
  - Proration only allowed on limited grounds (e.g., health related)

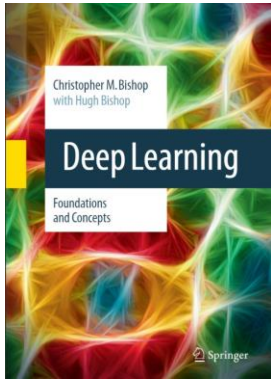
Textbooks and Readings

- Some books that you may use as reference (freely available online)
  - 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.
  - Chris Bishop, Pattern Recognition and Machine Learning (PRML), Springer, 2007.
  - Chris Bishop and Hugh Bishop, Deep Learning: Foundations and Concepts (DLFC), Springer, 2023.









- 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

Probabilistic Machine Learning

- Machine Learning primarily deals with
  - Predicting output $\boldsymbol{y}_*$ for new (test) inputs $\boldsymbol{x}_*$ given training data $(\boldsymbol{X}, \boldsymbol{y}) = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$
  - Generating new (synthetic) data given some training data $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$
- Probabilistic ML gives a natural way to solve both these tasks (with some advantages)
- Prediction: Learning the predictive distribution

$$p(y_* | x_*, \boldsymbol{X}, \boldsymbol{y})$$

Using this, we can not only get the mean but also the variance (uncertainty) of the predicted output y_*

- Generation: Learning a generative model of data

$$p(\boldsymbol{x}_* | \boldsymbol{X})$$

Can "sample" (simulate) from this distribution to generate new data

PML is about estimating these distributions accurately and efficiently

Estimating them exactly is hard in general but we can use approximations

Both are conditional distributions

- At its core, both problems require estimating the underlying distribution of data

Probabilistic Machine Learning

- With a probabilistic approach to ML, we can also easily incorporate "domain knowledge"

- Can specify our assumptions about data using suitable probability distributions over inputs/outputs, usually in the forms

Probability distribution of the output as a function of input

$$p(y_n|x_n, \theta)$$

Unknown parameters of this distribution

$$p(x_n|y_n, \theta)$$

Distribution of the input conditioned on its "label/output"

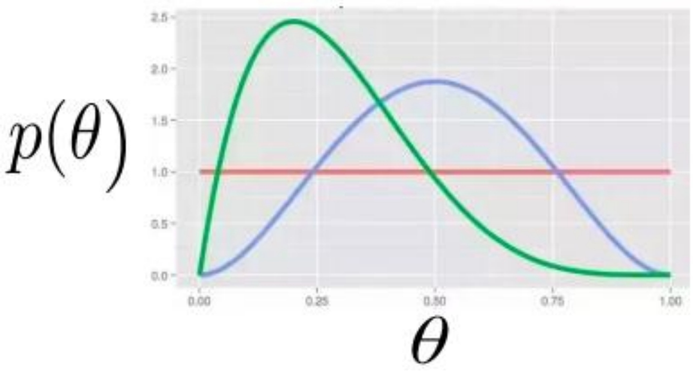
$$p(x_n|\theta)$$

Distribution of the inputs

- Can specify our assumptions about the unknowns θ using a "prior distribution"

Represents our belief about the unknown parameters before we see the data

$$p(\theta)$$



- After seeing some data $\mathcal{D}$, can update the prior into a posterior distribution $p(\theta|\mathcal{D})$

The Core of PML: Two Basic Rules of Probability

- Sum Rule (marginalization): Distribution of $\boldsymbol{a}$ considering for all possibilities of $\boldsymbol{b}$

If $\boldsymbol{b}$ is a discrete r.v.

$$p(a) = \sum_{b} p(a,b)$$

or

If $\boldsymbol{b}$ is a continuous r.v.

$$p(a) = \int p(a,b)db$$

- Product Rule

$$p(a,b) = p(a)p(b|a) = p(b)p(a|b)$$

- These two rules are the core of most of probabilistic/Bayesian ML
  - Bayes rule easily derived from the sum and product rules

$$p(b|a) = \frac{p(b)p(a|b)}{p(a)} = \frac{p(b)p(a|b)}{\int p(a,b)db}$$

Assuming $\boldsymbol{b}$ is a continuous r.v.

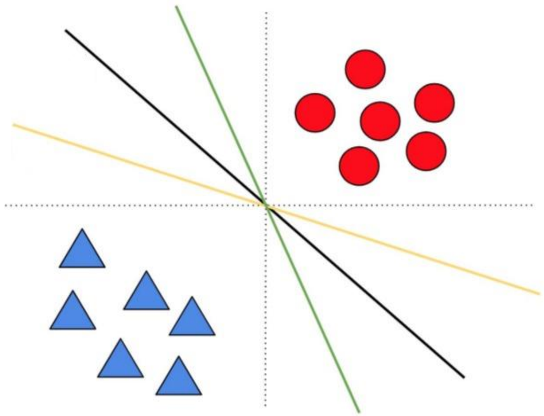
ML and Uncertainty
(and how PML handles uncertainty)

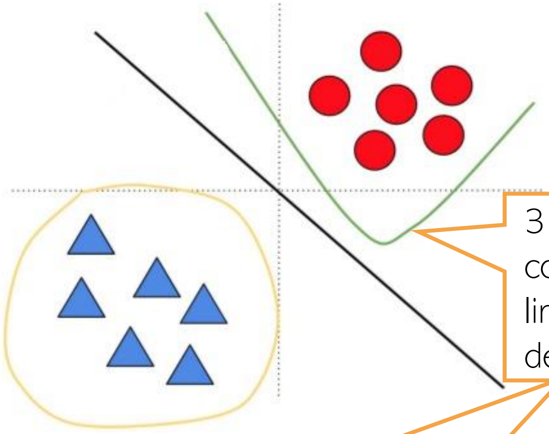
Uncertainty due to Limited Training Data

- Model/parameter 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 called epistemic uncertainty. Usually reducible
  - Vanishes with "sufficient" training data

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

Uncertainty due to Inherent Noise in Training Data

- 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

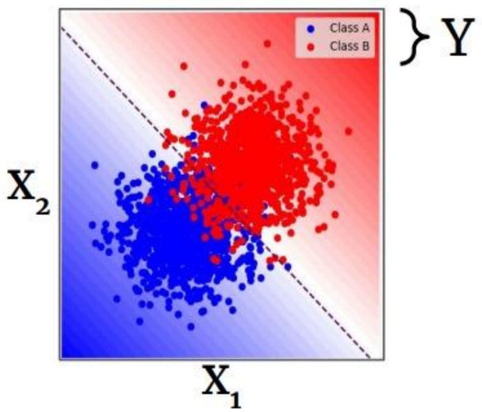


Image credit: Eric Nalisnick

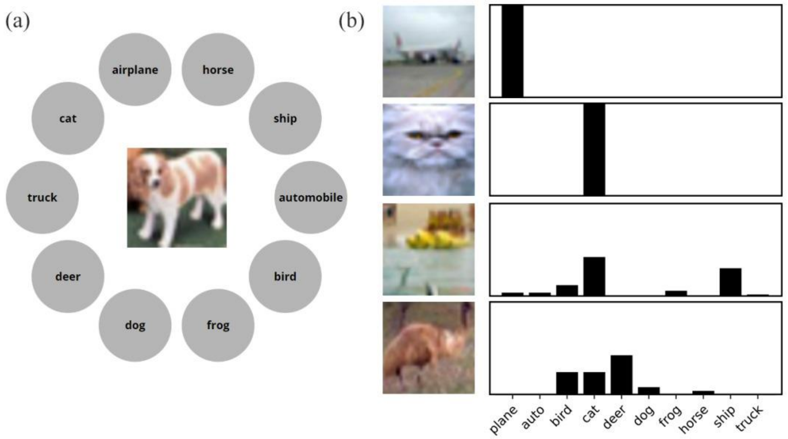


Image source: "Improving machine classification using human uncertainty measurements" (Battleday et al, 2021)

- 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

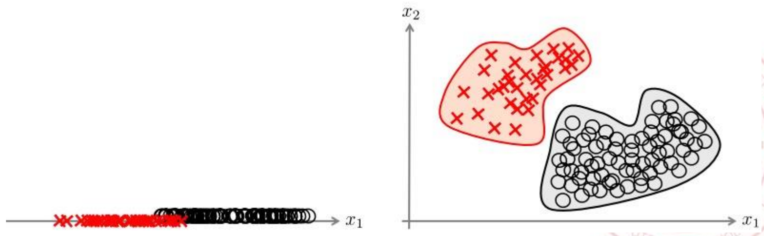


Image source: "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods" (H&W 2021)

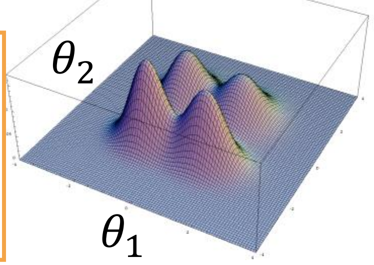
How to Estimate Uncertainty?

In this course, we will mostly focus on the Bayesian approach but other two approaches are also popular and will also be discussed

- **Uncertainty in parameters:** This can be estimated/quantified via mainly three ways:

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

A case of 2-dim θ

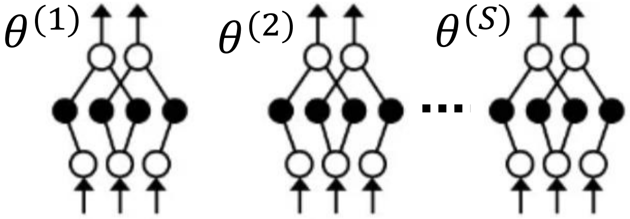


Bayesian way: Treat params as random variables and estimate their distribution conditioned on the given training data (a.k.a. posterior distribution)

Sampling multiple training sets and estimating the parameters from each training set

$$\{\hat{\theta}(\mathcal{D}') : \mathcal{D}' \sim p^*\}$$

Frequentist way: Treat params as fixed unknowns and estimate them using multiple datasets. This yields a set/distribution over the params (not a "posterior" but a distribution nevertheless!)



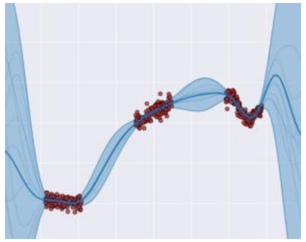
Ensemble: Train the same model with S different initializations or different subsets of the training data. Each run will give a different estimate, so we get a set of param estimates

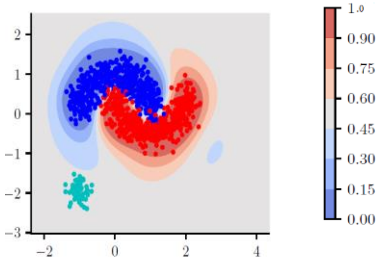
- **Uncertainty in predictions:** Usually estimated by computing and reporting the mean and variance of predictions made using many possible values of $\boldsymbol{\theta}$. Commonly reported as:

Predictive Distribution

$p(y_*|x_*, \mathcal{D})$

Can get both mean and variance/quantiles of the prediction





Sets/intervals of possible predictions

Predictive Uncertainty

- Information about uncertainty gives an idea about how much to trust a prediction
- It can also "guide" us in sequential decision-making:

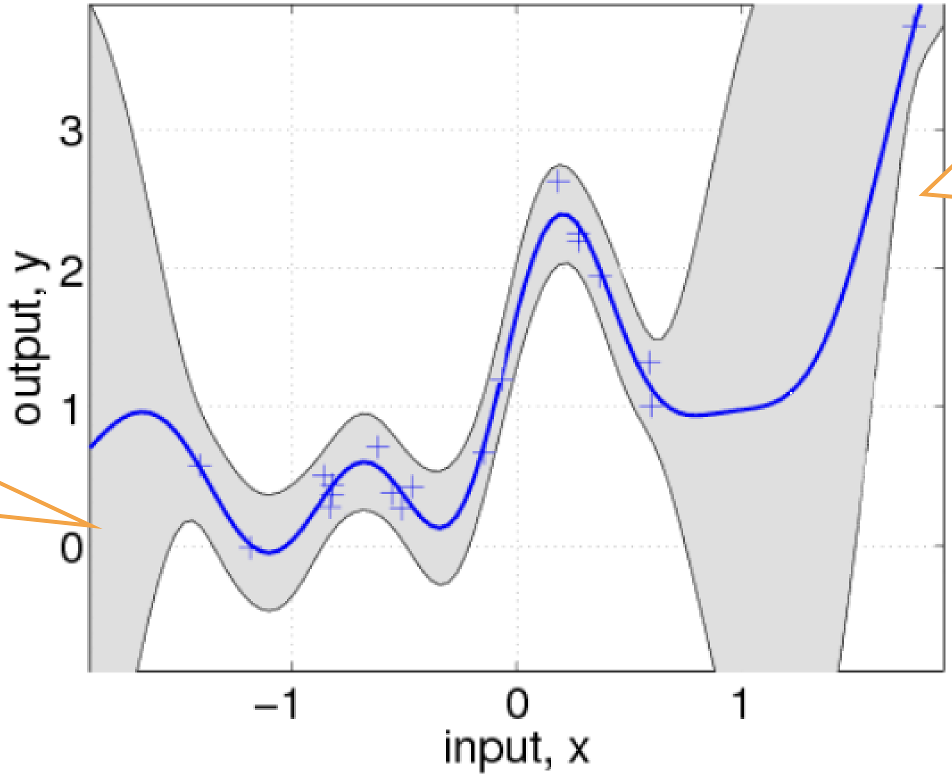
Test output

Test input

$$p(y_*|x_*, D) = \mathcal{N}(y_*|\mu_*, \sigma_*^2)$$

Training data

Blue curve is the mean of the function (learned so far using the available data), shaded region denotes the current predictive uncertainty



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

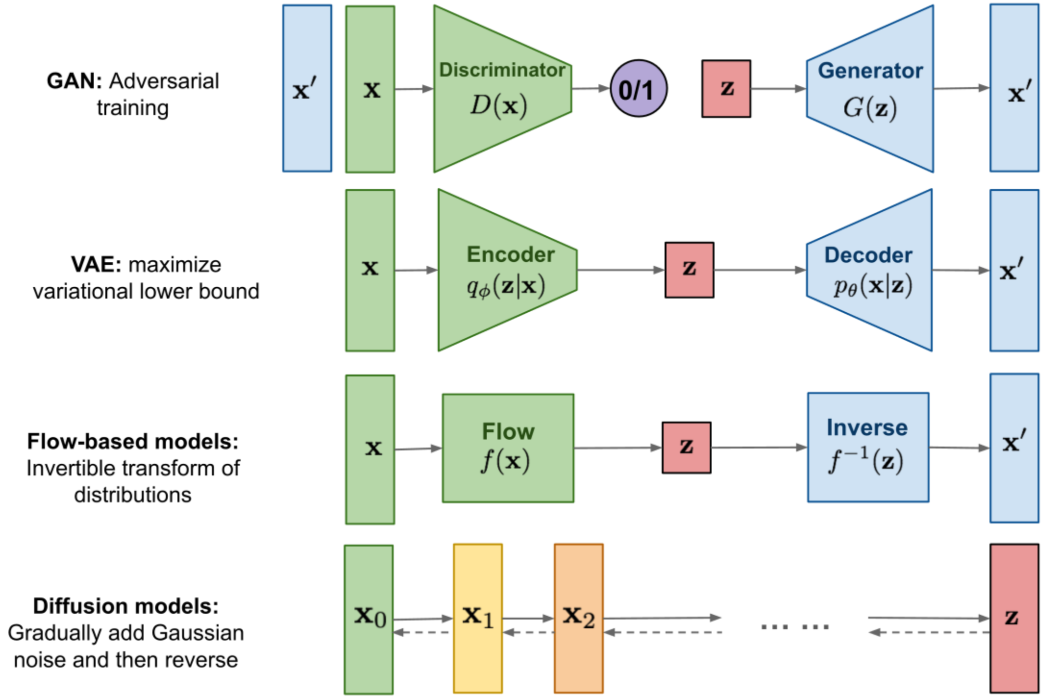
Uncertainty can help here: Acquire training inputs from regions where the function is most uncertain about its current predictions

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

Generative Models

- PML is not just about parameter/predictive uncertainty

- Generative models invariably are also probabilistic models



- Learning such models will also be a topic of study in this course

Figure credit: Lilian Weng

Probabilistic Modeling of Data: The Setup

- We are given some training data $\mathcal{D}$
- For supervised learning, $\mathcal{D}$ contains N input-label pairs $(\boldsymbol{x}_i, y_i)_{i=1}^N$
- For unsupervised learning, $\mathcal{D}$ contains N inputs $(\boldsymbol{x}_i)_{i=1}^N$
- Other settings are also possible (e.g., semi-sup., reinforcement learning, etc)
- Assume that the observations are generated by a probability distribution
  - For now, assume the form of the distribution to be known (e.g. a Gaussian)
- Parameters of this distribution, collectively denoted by $\boldsymbol{\theta}$ are unknown
- Our goal is to estimate the distribution (and thus $\boldsymbol{\theta}$) using training data
- Once the distribution is estimated, we can do things such as
  - Predict labels of new inputs, along with our confidence in these predictions
  - Generate new data with similar properties as training data
  - .. and a lot of other useful tasks, e.g., detecting outliers

Probabilistic Modeling of Data: The Setup

- We will denote the data distribution as $p_\theta(\mathcal{D})$ or $p(\mathcal{D}|\theta)$
- Assume that, conditioned on θ, observations are independently and identically distributed (i.i.d. assumption). Depending on the problem, this may look like:

Supervised generative model (both inputs and output are modeled using a distribution)

$$(\boldsymbol{x}_n, y_n) \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x}, y|\theta) \implies p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(\boldsymbol{x}_i, y_i|\theta)$$

Supervised discriminative model (only the output is modeled using a distribution); input is assumed "given" and not modeled

$$y_n \overset{\text{i.i.d.}}{\sim} p(y|\boldsymbol{x}, \theta) \implies p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \theta)$$

Unsupervised generative model (there are only inputs; no labels)

$$\boldsymbol{x}_n \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x}|\theta) \implies p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(\boldsymbol{x}_i|\theta)$$

- Assume that both training and test data come from the same distribution
  - This assumption, although standard, may be violated in real-world applications of ML and there are "adaptation" methods to handle that