Kernel Methods (contd) and Probabilistic Modeling of Data

CS771: Introduction to Machine Learning

Speeding-up Kernel Methods

- Kernels assume that

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$$

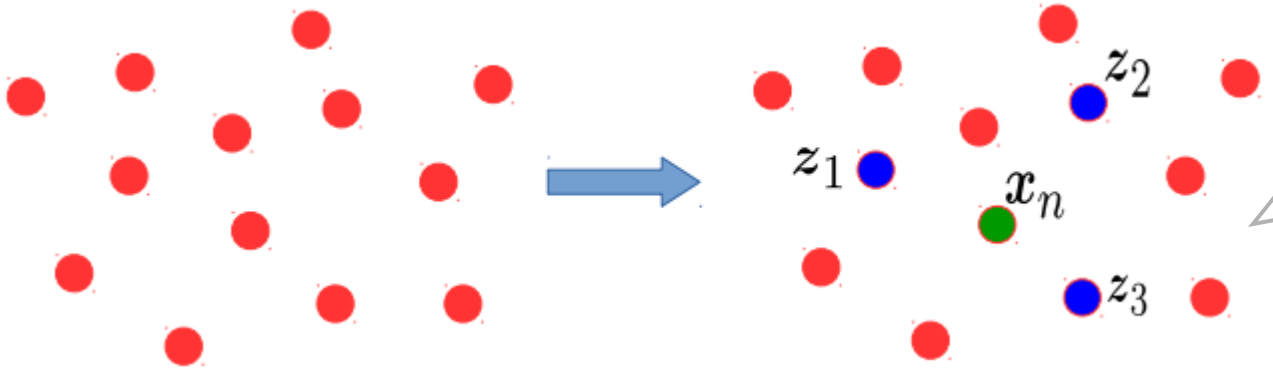
- Suppose for this kernel, we can get an L-dim feature vector $\psi(\boldsymbol{x})$ such that

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) \approx \psi(\boldsymbol{x}_n)^\top \psi(\boldsymbol{x}_m)$$

- Using features $\psi(\boldsymbol{x})$, we can learn a <u>linear model</u> with weights $\boldsymbol{w} \in \mathbb{R}^L$
- This model will be a good approximation to the kernelized model
- Training will be faster because no need to store and work with kernel matrices
- Prediction at test time will also be faster - we just need to compute $\boldsymbol{w}^\top \psi(\boldsymbol{x}_*)$
- Many ways to get such features $\psi(\boldsymbol{x})$ for standard kernels

Extracting Features using Kernels: Landmarks

- Suppose we choose a small set of L "landmark" inputs $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_L$ in the training data



Landmarks need not be actual inputs; can even be L learned locations in the input space

$$\psi(\boldsymbol{x}_n) = [k(\boldsymbol{z}_1, \boldsymbol{x}_n), k(\boldsymbol{z}_2, \boldsymbol{x}_n), k(\boldsymbol{z}_3, \boldsymbol{x}_n)] \in \mathbb{R}^3$$

- For each input $\boldsymbol{x}_n$, using a kernel k, define an L-dimensional feature vector as follows

$$\psi(\boldsymbol{x}_n) = [k(\boldsymbol{z}_1, \boldsymbol{x}_n), k(\boldsymbol{z}_2, \boldsymbol{x}_n), \ldots, k(\boldsymbol{z}_L, \boldsymbol{x}_n)] \in \mathbb{R}^L$$

- Can now apply a linear model on ψ representation (L-dimensional now) of the inputs
- This will be fast both at training as well as test time if L is small
- No need to kernelize the linear model while still reaping the benefits of kernels ☺

Extracting Feat. using Kernels: Random Features

- Many kernel functions* can be written as

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = \mathbb{E}_{\boldsymbol{w}\sim p(\boldsymbol{w})}[t_{\boldsymbol{w}}(\boldsymbol{x}_n) t_{\boldsymbol{w}}(\boldsymbol{x}_m)]$$

.. where $\boldsymbol{t_w}(.)$ is a function with params $\boldsymbol{w} \in \mathbb{R}^D$ with $\boldsymbol{w}$ drawn from some distr. $\boldsymbol{p(w)}$

- Example: For the RBF kernel, $\boldsymbol{t_w}(.)$ is cosine func. and $\boldsymbol{p(w)}$ is zero mean Gaussian

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \mathbb{E}_{\boldsymbol{w}\sim p(\boldsymbol{w})}[\cos(\boldsymbol{w}^\top \boldsymbol{x}_n) \cos(\boldsymbol{w}^\top \boldsymbol{x}_m)]$$

- Given $\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_L}$ from $\boldsymbol{p(w)}$, using Monte-Carlo approx. of above expectation

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) \approx \frac{1}{L} \sum_{\ell=1}^{L} \cos(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n) \cos(\boldsymbol{w}_\ell^\top \boldsymbol{x}_m) = \psi(\boldsymbol{x}_n)^\top \psi(\boldsymbol{x}_m)$$

.. where $\psi(\boldsymbol{x}_n) = \frac{1}{\sqrt{L}}[\cos(\boldsymbol{w}_1^\top \boldsymbol{x}_n), \ldots, \cos(\boldsymbol{w}_L^\top \boldsymbol{x}_n)]$ is an L-dim vector

- Can apply a linear model on this L-dim rep. of the inputs (no need to kernelize)

*Random Features for Large-Scale Kernel Machines (Ben and Retch, NIPS 2007. Note: This paper actually won the test-of-time award at NIPS 2017)

Learning with Kernels: Some Aspects

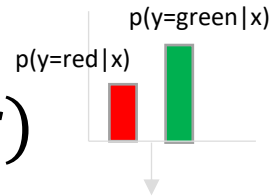
- Storage/computational efficiency can be a bottleneck when using kernels
- During training, need to compute and store the $N \times N$ kernel matrix $\boldsymbol{K}$ in memory
- Need to store training data (or at least support vectors in case of SVMs) at test time
- Test time can be slow: $O(N)$ cost to compute a quantity like $\sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}_*)$
- Approaches like landmark and random features can be used to speed up
- Choice of the right kernel is also very important
- Some kernels (e.g., RBF) work well for many problems but hyperparameters of the kernel function may need to be tuned via cross-validation
- Quite a bit of research on learning the right kernel from data
  - Learning a combination of multiple kernels (Multiple Kernel Learning)
  - Bayesian kernel methods (e.g., Gaussian Processes) can learn the kernel hyperparameters from data (thus can be seen as learning the kernel)
  - Deep Learning can also be seen as learning the kernel from data (more on this later)

Also, a lot of recent work on connections between kernel methods and deep learning

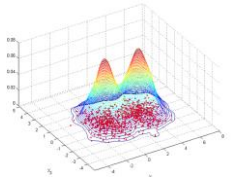
Probabilistic Modeling of Data

The Probabilistic Approach to ML

- Many ML problems can be seen as estimating a probability distribution/density
- Sup. Learning: Given labelled data $(X, y) = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, estimate $p(y|\boldsymbol{x})$



- Unsup. Learning: Given unlabelled data $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, estimate $p(\boldsymbol{x})$



- We estimate these using the given training data
  - These distributions will have some parameters $\boldsymbol{\theta}$ (to be estimated)
  - These distributions will typically have a known form (which we will assume, e.g., Gaussian), but sometimes not (i.e., the form itself may also need to be estimated)

Distribution of test data conditioned on the training data and any other relevant quantities

- Once these are estimated, we can compute predictive distributions, e.g.,
  - Sup. Learning: Given a new test input $\boldsymbol{x}_*$, what is $p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})$, or mean/variance of y_*?
  - Unsup. Learning: Given a new test input $\boldsymbol{x}_*$, what is $p(\boldsymbol{x}_*|\boldsymbol{X})$?

Getting Started: A Simple Setting

E.g., outcomes of N coin tosses, or heights of N students in a class

- Assume we are given N observations $\boldsymbol{y} = \{y_1, y_2, \dots, y_N\}$
- Assume these are generated from a probability model (a distribution)

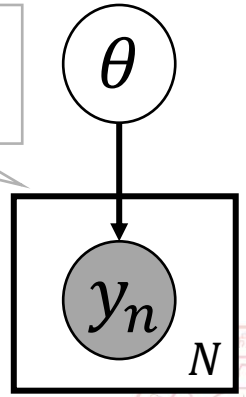
$$y_n \sim p(y|\theta) \qquad \forall n$$

(assumed independently & identically distributed (i.i.d.))

- Assume the form of $p(y|\theta)$ to be known (e.g., Bernoulli or Gaussian) and parameters $\boldsymbol{\theta}$ of this distribution to be unknown
- Estimating $\boldsymbol{\theta}$ here means we are estimating the distribution
- We can perform estimation of $\boldsymbol{\theta}$ in two ways
  - Its single best/optimal value (called "point estimate")
  - A set/distribution of likely values
- Finally, we may be interested in the predictive distribution $p(y_*|\boldsymbol{y})$ as well

Such a diagram is called a plate diagram

Shaded nodes mean the value of the variable is observed, white node means the value is unobserved



New test observation

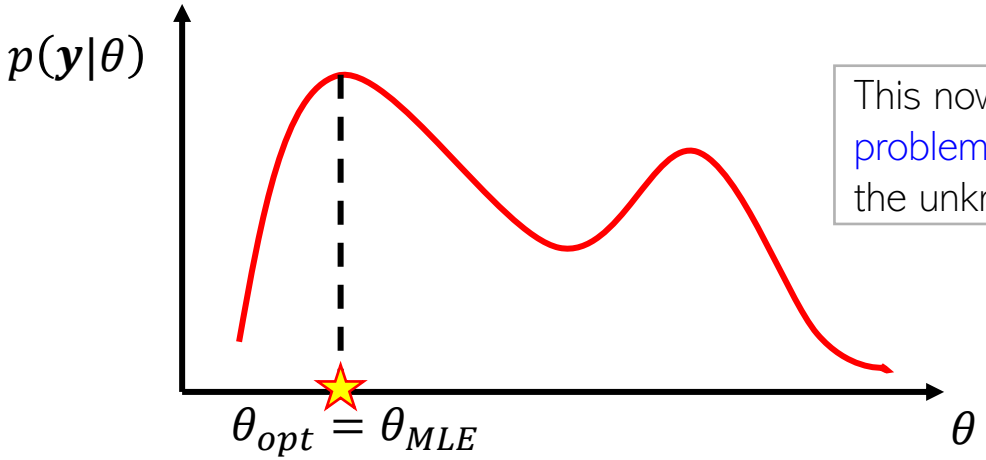
Parameter Estimation in Probabilistic Models

- Since data is assumed to be i.i.d., we can write down its total probability as

$$p(\boldsymbol{y}|\theta) = p(y_1, y_2, \dots, y_N|\theta) = \prod_{n=1}^{N} p(y_n|\theta)$$

- $p(\boldsymbol{y}|\theta)$ called "likelihood" - probability of observed data as a function of params $\boldsymbol{\theta}$
- We wish to find the "best" $\boldsymbol{\theta}$, given observed data $\boldsymbol{y}$
- One notion of "best" is to find $\boldsymbol{\theta}$ which maximizes the likelihood

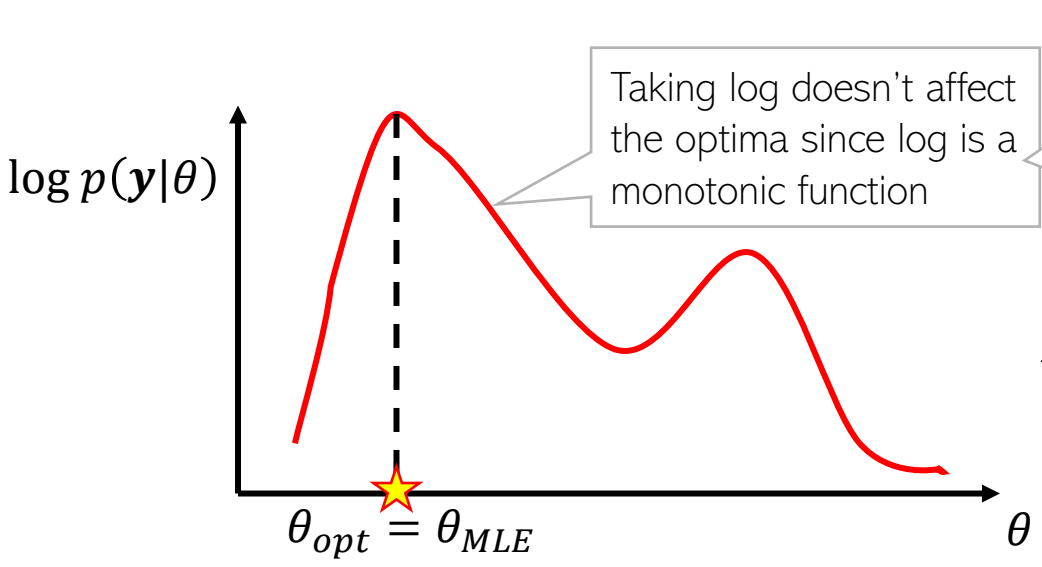
Basically, which value of $\boldsymbol{\theta}$ makes the observed data most probable under the assumed distribution $p(\boldsymbol{y}|\theta)$



This now is an optimization problem essentially ($\boldsymbol{\theta}$ being the unknown)

Maximum Likelihood Estimation (MLE)

- The goal in MLE is to find the optimal $\boldsymbol{\theta}$ by maximizing the likelihood
- In practice, we maximize the log of the likelihood (log-likelihood in short)



Taking log doesn't affect the optima since log is a monotonic function

Leads to simpler algebra/calculus, and also yields better numerical stability when implementing it on computer (dealing with log of probabilities)

$$LL(\theta) = \log p(\boldsymbol{y}|\theta) = \log \prod_{n=1}^{N} p(y_n|\theta)$$

$$= \sum_{n=1}^{N} \log p(y_n|\theta)$$

- Thus the MLE problem is

$$\boxed{\theta_{MLE} = \underset{\theta}{\operatorname{argmax}}\, LL(\theta) = \underset{\theta}{\operatorname{argmax}} \sum_{n=1}^{N} \log p(y_n|\theta)}$$

- This is now an optimization (maximization problem)

Maximum Likelihood Estimation (MLE)

Negative Log-Likelihood (NLL)

- The MLE problem can also be easily written as a <u>minimization</u> problem

$$\theta_{MLE} = \underset{\theta}{\text{argmax}} \sum_{n=1}^{N} \log p(y_n|\theta) = \underset{\theta}{\text{argmin}} \sum_{n=1}^{N} -\log p(y_n|\theta)$$

- Thus MLE can also be seen as minimizing the negative log-likelihood (NLL)

$$\boxed{\theta_{MLE} = \underset{\theta}{\text{argmin}}\, NLL(\theta)}$$

Indeed. It may overfit. Several ways to prevent it: Use regularizer or other strategies to prevent overfitting. Alternatives, use "prior" distributions on the parameters θ that we are trying to estimate (which will kind of act as a regularizer as we will see shortly)

Such priors have various other benefits as we will see later

- NLL is analogous to a loss function
  - The negative log-lik $(-\log p(y_n|\theta))$ is akin to the loss on each data point
- Thus doing MLE is akin to <u>minimizing training loss</u>

Does it mean MLE could overfit? If so, how to prevent this?

MLE: An Example

- Consider a sequence of N coin toss outcomes (observations)
- Each observation y_n is a binary random variable. Head: $y_n = 1$, Tail: $y_n = 0$
- Each y_n is assumed generated by a **Bernoulli distribution** with param $\theta \in (0,1)$

Probability of a head

$$p(y_n|\theta) = \text{Bernoulli}(y_n|\theta) = \theta^{y_n}(1-\theta)^{1-y_n}$$

- Here θ the unknown param (probability of head). Want to estimate it using MLE
- Log-likelihood: $\sum_{n=1}^{N} \log p(y_n|\theta) = \sum_{n=1}^{N} [y_n \log \theta + (1-y_n)\log(1-\theta)]$
- Maximizing log-lik (or minimizing NLL) w.r.t. θ will give a closed form expression

Take deriv. set it to zero and solve. Easy optimization

I tossed a coin 5 times – gave 1 head and 4 tails. Does it means $\theta = 0.2$?? The MLE approach says so. What is I see 0 head and 5 tails. Does it mean $\theta = 0$?

$$\theta_{MLE} = \frac{\sum_{n=1}^{N} y_n}{N}$$

Thus MLE solution is simply the fraction of heads! ☺ Makes intuitive sense!

Indeed – if you want to trust MLE solution. But with small number of training observations, MLE may overfit and may not be reliable. We will soon see better alternatives that use prior distributions!

MLE and Its Shortcomings..

- MLE finds parameter values s that make the observed data most probable

Log-likelihood

Neg. log-likelihood (NLL)

$$\theta_{MLE} = \underset{\theta}{\text{argmax}} \sum_{n=1}^{N} \log p(y_n|\theta) = \underset{\theta}{\text{argmin}} \sum_{n=1}^{N} -\log p(y_n|\theta)$$

- No provision to control overfitting (MLE is just like minimizing training loss)

- How do we regularize probabilistic models in a principled way?

- Also, MLE gives only a single "best" answer ("point estimate")
  - .. and it may not be very reliable, especially when we have very little data
  - Desirable: Report a <u>probability distribution</u> over the learned params <u>instead of point est</u>

This distribution can give us a sense about the <u>uncertainty</u> in the parameter estimate

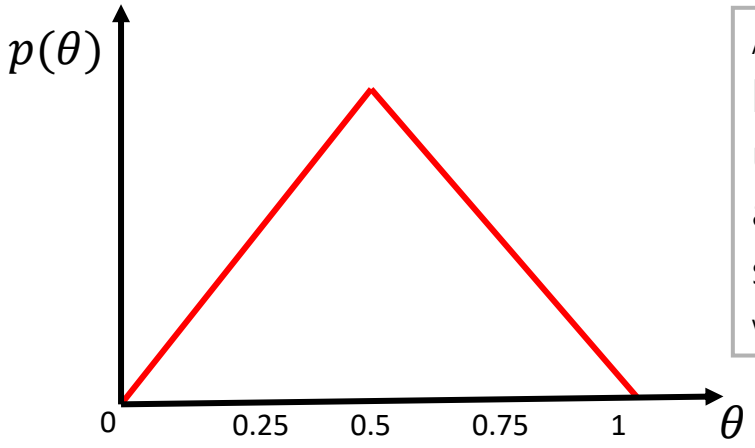
- Prior distributions provide a nice way to accomplish such things!

Priors

Before observing any data

- Can specify our prior belief about likely param values via a prob. dist., e.g.,

This is a rather simplistic/contrived prior. ☺ Just to illustrate the basic idea. We will see more concrete examples of priors shortly. Also, the prior usually depends (assumed conditioned on) on some fixed/learnable hyperparameters (say some α and β , and written as $p(\theta|\alpha,\beta)$



A possible prior for the coin bias estimation problem. The unknown θ is being treated as a random variable, not simply a fixed unknown as we treated it as in MLE

- Once we observe the data $\boldsymbol{y}$, apply Bayes rule to update prior into posterior

Prior

Likelihood

Posterior

$$p(\theta|\boldsymbol{y}) = \frac{p(\theta)p(\boldsymbol{y}|\theta)}{p(\boldsymbol{y})}$$

Marginal likelihood

Note: Marginal lik. is hard to compute in general as it requires a summation or integral which may not be easy (will briefly look at this in CS771, although will stay away going too deep in this course – CS772 does that in more detail)

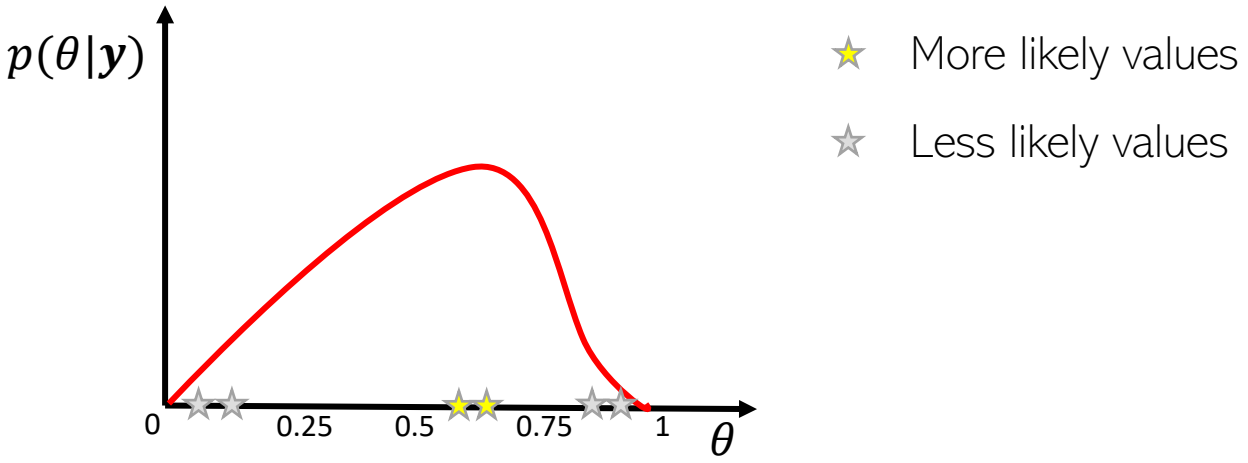
- Two ways now to report the answer:
  - Report the maxima (mode) of the posterior: $\arg\max_{\theta} p(\theta|\boldsymbol{y})$
  - Report the full posterior (and its properties, e.g., mean, mode, variance, quantiles, etc)

Maximum-a-posteriori (MAP) estimation

Fully Bayesian inference

Posterior

- Posterior distribution tells us how probable different parameter values are <u>after</u> we have observed some data
- Height of posterior at each value gives the posterior probability of that value



- Can think of the posterior as a "hybrid" obtained by combining information from the likelihood and the prior

Maximum-a-Posteriori (MAP) Estimation

- The MAP estimation approach reports the maxima/mode of the posterior

$$\theta_{MAP} = \arg\max_{\theta} p(\theta|y) = \arg\max_{\theta} \log p(\theta|y) = \arg\max_{\theta} \log \frac{p(\theta)p(\boldsymbol{y}|\theta)}{p(\boldsymbol{y})}$$

- Since $p(y)$ is constant w.r.t. θ, the above simplifies to

$$\theta_{MAP} = \arg\max_{\theta} [\log p(y|\theta) + \log p(\theta)]$$

$$= \arg\min_{\theta} [-\log p(y|\theta) - \log p(\theta)]$$

$$\boxed{\theta_{MAP} = \arg\min_{\theta} [NLL(\theta) - \log p(\theta)]}$$

The NLL term acts like the training loss and the (negative) log-prior acts as regularizer. Keep in mind this analogy. ☺

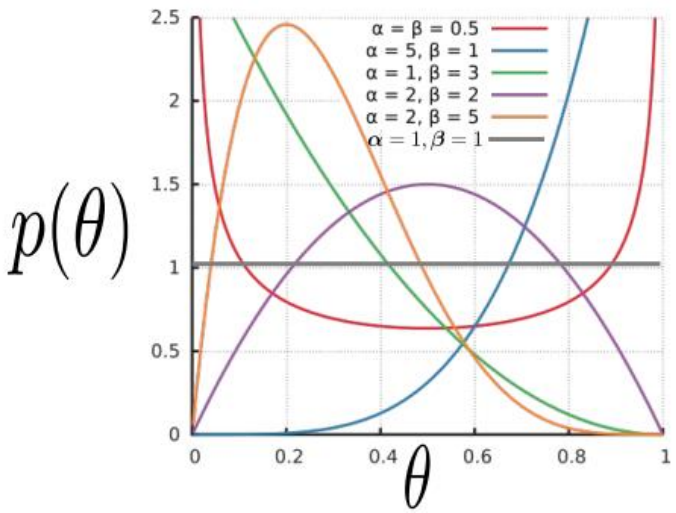
- Same as MLE with an extra log-prior-distribution term (acts as a regularizer) ☺
- If the prior is absent or <u>uniform</u> (all values equally likely a prior) then MAP=MLE

MAP Estimation: An Example

- Let's again consider the coin-toss problem (estimating the bias of the coin)
- Each likelihood term is Bernoulli

$$p(y_n|\theta) = \text{Bernoulli}(y_n|\theta) = \theta^{y_n}(1-\theta)^{1-y_n}$$

- Also need a prior since we want to do MAP estimation
- Since $\theta \in (0,1)$, a reasonable choice of prior for θ would be Beta distribution



$$p(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

The gamma function

Using $\alpha = 1$ and $\beta = 1$ will make the Beta prior a uniform prior

α and β (both non-negative reals) are the two hyperparameters of this Beta prior

Can set these based on intuition, cross-validation, or even learn them

MAP Estimation: An Example (Contd)

- The log posterior for the coin-toss model is log-lik + log-prior

$$LP(\theta) = \sum_{n=1}^{N} \log p(y_n|\theta) + \log p(\theta|\alpha,\beta)$$

- Plugging in the expressions for Bernoulli and Beta and ignoring any terms that don't depend on $\boldsymbol{\theta}$, the log posterior simplifies to

$$LP(\theta) = \sum_{n=1}^{N} [y_n \log \theta + (1-y_n)\log(1-\theta)] + (\alpha-1)\log\theta + (\beta-1)\log(1-\theta)$$

- Maximizing the above log post. (or min. of its negative) w.r.t. $\boldsymbol{\theta}$ gives

$$\theta_{MAP} = \frac{\sum_{n=1}^{N} y_n + \alpha - 1}{N + \alpha + \beta - 2}$$

Using $\alpha = 1$ and $\beta = 1$ gives us the same solution as MLE

Recall that $\alpha = 1$ and $\beta = 1$ for Beta distribution is in fact equivalent to a uniform prior (hence making MAP equivalent to MLE)

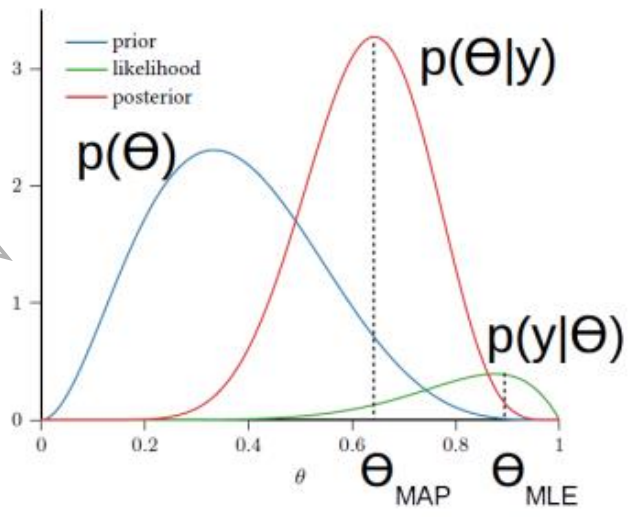
Prior's hyperparameters have an interesting interpretation. Can think of $\alpha - 1$ and $\beta - 1$ as the number of heads and tails, respectively, before starting the coin-toss experiment (akin to "pseudo-observations")

Such interpretations of prior's hyperparameters as being "pseudo-observations" exist for various other prior distributions as well (in particular, distributions belonging to "exponential family" of distributions

Fully Bayesian Inference

- MLE/MAP only give us a point estimate of $\boldsymbol{\theta}$

MAP estimate is more robust than MLE (due to the regularization effect) but the estimate of uncertainty is missing in both approaches – both just return a single "optimal" solution by solving an optimization problem



Interesting fact to keep in mind: Note that the use of the prior is making the MLE solution move towards the prior (MAP solution is kind of a "compromise between MLE solution of the mode of the prior) ☺

Fully Bayesian inference

- If we want more than just a point estimate, we can compute the full posterior

$$p(\theta|\boldsymbol{y}) = \frac{p(\theta)p(\boldsymbol{y}|\theta)}{p(\boldsymbol{y})}$$

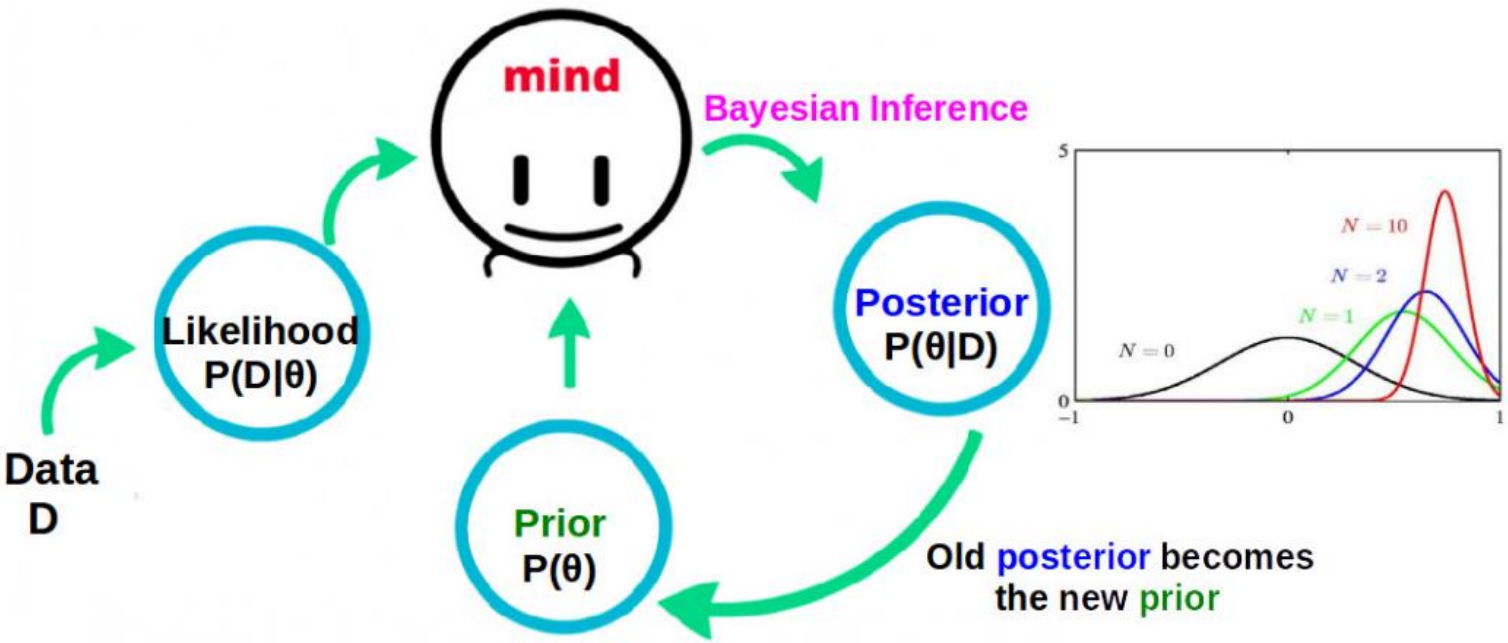
Computable analytically only when the prior and likelihood are "friends" with each other (i.e., they form a conjugate pair of distributions (distributions from exponential family have conjugate priors

An example: Bernoulli and Beta are conjugate. Will see some more such pairs

In other cases, the posterior needs to be approximated (will see 1-2 such cases in this course; more detailed treatment in the advanced course on probabilistic modeling and inference)

"Online" Nature of Bayesian Inference

- Fully Bayesian inference fits naturally into an "online" learning setting



Also, the posterior becomes more and more "concentrated" as the number of observations increases. For very large N, you may expect it to be peak around the MLE solution

- Our belief about θ keeps getting updated as we see more and more data

Fully Bayesian Inference: An Example

Also, if you get more observations, you can treat the current posterior as the new prior and obtain a new posterior using these extra observations

Posterior is the same distribution as the prior (both Beta), just with updated hyperparameters (property when likelihood and prior are conjugate to each other)

- Let's again consider the coin-toss problem
- Bernoulli likelihood: $p(y_n|\theta) = \text{Bernoulli}(y_n|\theta) = \theta^{y_n}(1-\theta)^{1-y_n}$
- Beta prior: $p(\theta) = \text{Beta}(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$
- The posterior can be computed as

Number of heads (N_1)

Number of tails (N_0)

$$\theta^{\sum_{n=1}^N y_n}(1-\theta)^{N-\sum_{n=1}^N y_n}$$

$$p(\theta|\boldsymbol{y}) = \frac{p(\theta)p(\boldsymbol{y}|\theta)}{p(\boldsymbol{y})} = \frac{p(\theta)\prod_{n=1}^N p(y_n|\theta)}{p(\boldsymbol{y})} = \frac{\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}\prod_{n=1}^N \theta^{y_n}(1-\theta)^{1-y_n}}{\int \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}\prod_{n=1}^N \theta^{y_n}(1-\theta)^{1-y_n}d\theta}$$

This is the numerator integrated/marginalized over $\theta : p(\mathbf{y}) = \int p(\theta,\boldsymbol{y})d\theta = \int p(\theta)p(\boldsymbol{y}|\theta)d\theta$

In general, hard but with conjugate pairs of prior and likelihood, we don't need to compute this, as we will see in this example ☺

Parts coming from the numerator, which consist of θ terms. We have ignored other constants in the numerator, and the whole denominator which is also constant w.r.t. θ

$$\propto \theta^{\alpha+N_1-1}(1-\theta)^{\beta+N_0-1}$$

Aha! This is nothing but $\text{Beta}(\theta|\alpha+N_1,\beta+N_0)$

This, of course, is not always possible but only in simple cases like this

Found the posterior just by simple inspection without having to calculate the constant of proportionality ☺

Conjugacy

- Many pairs of distributions are conjugate to each other
  - Bernoulli (likelihood) + Beta (prior) ⇒ Beta posterior
  - Binomial (likelihood) + Beta (prior) ⇒ Beta posterior
  - Multinomial (likelihood) + Dirichlet (prior) ⇒ Dirichlet posterior
  - Poisson (likelihood) + Gamma (prior) ⇒ Gamma posterior
  - Gaussian (likelihood) + Gaussian (prior) ⇒ Gaussian posterior
  - and many other such pairs ..
- Tip: If two distr are conjugate to each other, their functional forms are similar
  - Example: Bernoulli and Beta have the forms

$$\text{Bernoulli}(y|\theta) = \theta^{y}\,(1-\theta)^{1-y}$$

$$\text{Beta}(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\,\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

This is why, when we multiply them while computing the posterior, the exponents get added and we get the same form for the posterior as the prior but with just updated hyperparameter. Also, we can identify the posterior and its hyperparameters simply by inspection

Probabilistic Models: Making Predictions

- Having estimated $\boldsymbol{\theta}$, we can now use it to make predictions
- Prediction entails computing the predictive distribution of a new observation, say $\boldsymbol{y}_*$

For example, PMF of the label of a new test input in classification

$$p(y_*|\boldsymbol{y}) = \int p(y_*, \theta|\boldsymbol{y})d\theta$$

Marginalizing over the unknown θ

Conditional distribution of the new observation, given past observations

$$= \int p(y_*|\theta, \boldsymbol{y})p(\theta|\boldsymbol{y})d\theta$$

Decomposing the joint using chain rule

$$= \int p(y_*|\theta)p(\theta|\boldsymbol{y})d\theta$$

Assuming i.i.d. data, given θ, y_* does not depend on $\boldsymbol{y}$

- When doing MLE/MAP, we approximate the posterior $p(\theta|\boldsymbol{y})$ by a single point θ_{opt}

$$p(y_*|\boldsymbol{y}) = \int p(y_*|\theta)p(\theta|\boldsymbol{y})d\theta \approx p(y_*|\theta_{opt})$$

A "plug-in prediction" (simply plugged in the single best estimate we had)

- When doing fully Bayesian estimation, getting the predictive dist. will require computing

$$p(y_*|\boldsymbol{y}) = \int p(y_*|\theta)p(\theta|\boldsymbol{y})d\theta$$

$\mathbb{E}_{p(\theta|\boldsymbol{y})}[p(y_*|\theta)]$

This computes the predictive distribution by averaging over the full posterior – basically calculate $p(y_*|\theta)$ for each possible θ, weighs it by how likely this θ is under the posterior $p(\theta|\boldsymbol{y})$, and sum all such posterior weighted predictions. Note that not each value of theta is given equal importance here in the averaging

Probabilistic Models: Making Predictions (Example)

- For coin-toss example, let's compute probability of the $(N+1)^{th}$ toss showing head

- This can be done using the MLE/MAP estimate, or using the full posterior

$$\theta_{MLE} = \frac{N_1}{N} \qquad \theta_{MAP} = \frac{N_1 + \alpha - 1}{N + \alpha + \beta - 2} \qquad p(\theta|\boldsymbol{y}) = \text{Beta}(\theta|\alpha + N_1, \beta + N_0)$$

- Thus for this example (where observations are assumed to come from a Bernoulli)

MLE prediction: $p(y_{N+1} = 1|\boldsymbol{y}) = \int p(y_{N+1} = 1|\theta)p(\theta|\boldsymbol{y})d\theta \approx p(y_{N+1} = 1|\theta_{MLE}) = \theta_{MLE} = \frac{N_1}{N}$

MAP prediction: $p(y_{N+1} = 1|\boldsymbol{y}) = \int p(y_{N+1} = 1|\theta)p(\theta|\boldsymbol{y})d\theta \approx p(y_{N+1} = 1|\theta_{MAP}) = \theta_{MAP} = \frac{N_1 + \alpha - 1}{N + \alpha + \beta - 2}$

Fully Bayesian: $p(y_{N+1} = 1|\boldsymbol{y}) = \int p(y_{N+1} = 1|\theta)p(\theta|\boldsymbol{y})d\theta = \int \theta p(\theta|\boldsymbol{y})d\theta = \int \theta \text{Beta}(\theta|\alpha + N_1, \beta + N_0)d\theta = \frac{N_1 + \alpha}{N + \alpha + \beta}$

Again, keep in mind that the posterior weighted averaged prediction used in the fully Bayesian case would usually not be as simple to compute as it was in this case. We will look at some hard cases later

Expectation of θ under the Beta posterior that we computed using fully Bayesian inference

Probabilistic Modeling: A Summary

- Likelihood corresponds to a loss function; prior corresponds to a regularizer
- Can choose likelihoods and priors based on the nature/property of data/parameters
- MLE estimation = unregularized loss function minimization
- MAP estimation = regularized loss function minimization
- Allows us to do fully Bayesian learning (learning the full distribution of the parameters)
- Makes robust predictions by posterior averaging (rather than using point estimate)
- Many other benefits, such as
  - Estimate of confidence in the model's prediction (useful for doing Active Learning)
  - Can do automatic model selection, hyperparameter estimation, handle missing data, etc.
  - Formulate latent variable models
  - .. and many other benefits (a proper treatment deserves a separate course, but we will see some of these in this course, too)