Probabilistic Machine Learning (1): Some Basics of Probability

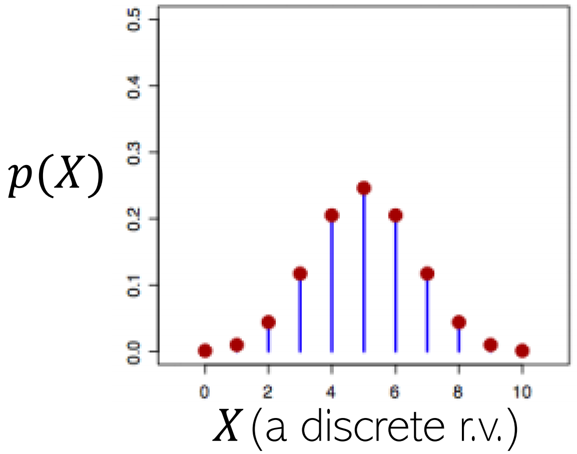
CS771: Introduction to Machine Learning

Piyush Rai

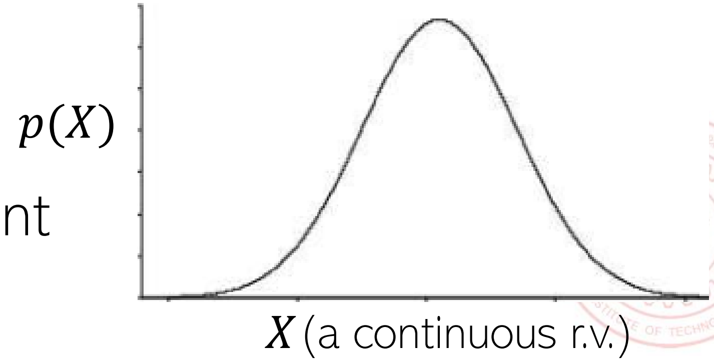
Some Probability Basics

Random Variables

- Informally, a random variable (r.v.) X denotes possible outcomes of an event
- Can be discrete (i.e., finite many possible outcomes) or continuous
- Some examples of discrete r.v.
  - $X \in \{0,1\}$ denoting outcomes of a coin-toss
  - $X \in \{1,2,\ldots,6\}$ denoting outcome of a dice roll



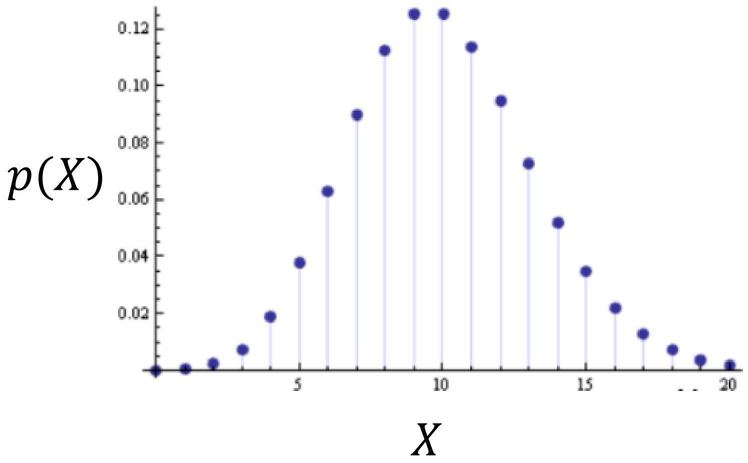
- Some examples of continuous r.v.
  - $X \in (0,1)$ denoting the bias of a coin
  - $X \in \mathbb{R}$ denoting heights of students in CS771
  - $X \in \mathbb{R}$ denoting time to get to your hall from the department



Discrete Random Variables

- For a discrete r.v. X, $p(x)$ denotes $p(X = x)$ - probability that $X = x$
- $p(X)$ is called the probability mass function (PMF) of r.v. X
    - $p(x)$ or $p(X = x)$ is the value of the PMF at x

$$p(x) \geq 0$$
$$p(x) \leq 1$$
$$\sum_x p(x) = 1$$



Continuous Random Variables

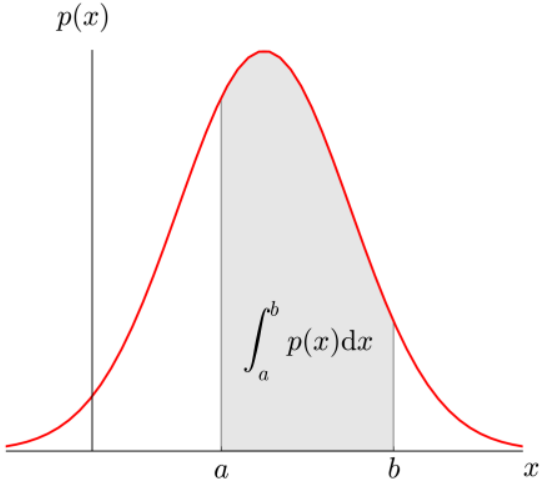
- For a continuous r.v. X, a *probability* $p(X = x)$ or $p(x)$ is meaningless
- For cont. r.v., we talk in terms of prob. within an interval $X \in (x, x + \delta x)$
  - $p(x)\delta x$ is the prob. that $X \in (x, x + \delta x)$ as $\delta x \to 0$
  - $p(x)$ is the probability density at $X = x$

Yes, probability density at a point x can very well be larger than 1. The integral however must be equal to 1

$$p(x) \geq 0$$

$$\cancel{p(x) \leq 1}$$

$$\int p(x)dx = 1$$



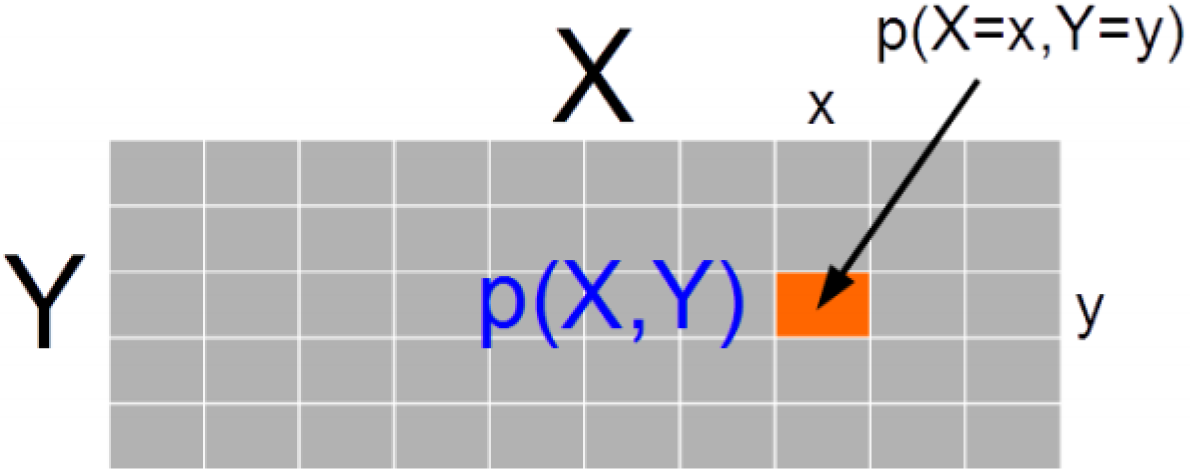
A word about notation

- $p(.)$ can mean different things depending on the context
- $p(X)$ denotes the distribution (PMF/PDF) of an r.v. $\boldsymbol{X}$
- $p(X = x)$ or $p_X(x)$ or simply $p(x)$ denotes the prob. or prob. density at value x
  - Actual meaning should be clear from the context (but be careful)
- Exercise same care when $p(.)$ is a specific distribution (Bernoulli, Gaussian, etc.)
- The following means generating a random sample from the distribution $p(X)$

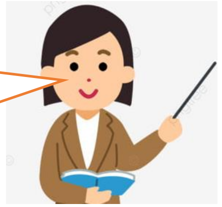
$$x \sim p(X)$$

Joint Probability Distribution

- Joint prob. dist. $p(X, Y)$ models probability of co-occurrence of two r.v. X, Y
- For discrete r.v., the joint PMF $p(X, Y)$ is like a table (that sums to 1)



For 3 r.v.'s, we will likewise have a "cube" for the PMF. For more than 3 r.v.'s too, similar analogy holds

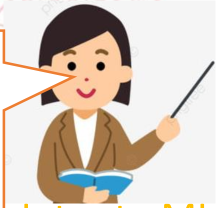


$$\sum_x \sum_y p(X = x, Y = y) = 1$$

- For two continuous r.v.'s X and Y, we have joint PDF $p(X, Y)$

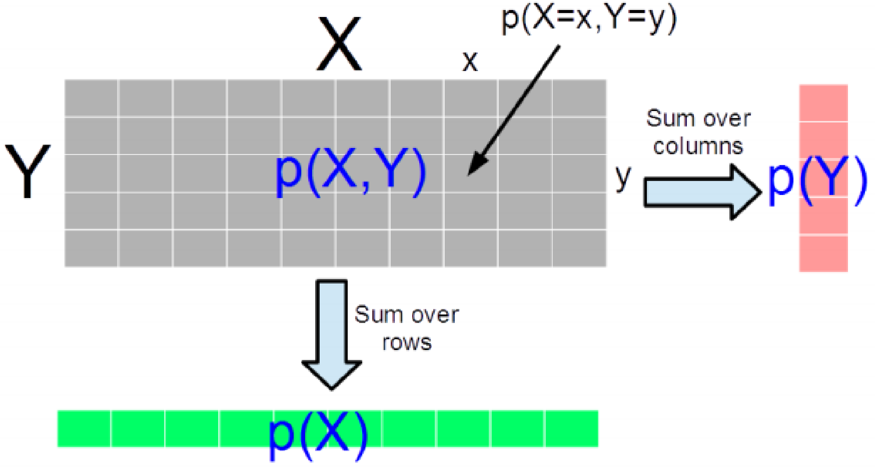
$$\int_x \int_y p(X = x, Y = y) dxdy = 1$$

For more than two r.v.'s, we will likewise have a multi-dim integral for this property



Marginal Probability Distribution

- Consider two r.v.'s X and Y (discrete/continuous – both need not of same type)
- Marg. Prob. is PMF/PDF of one r.v. accounting for all possibilities of the other r.v.
- For discrete r.v.'s, $p(X) = \sum_y p(X, Y = y)$ and $p(Y) = \sum_x p(X = x, Y)$
- For discrete r.v. it is the sum of the PMF table along the rows/columns



The definition also applied for two sets of r.v.'s and marginal of one set of r.v.'s is obtained by summing over all possibilities of the second set of r.v.'s

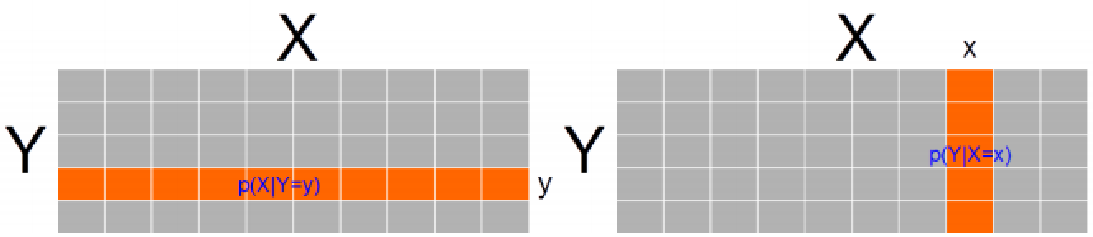
For discrete r.v.'s, marginalization is called summing over, for continuous r.v.'s, it is called "integrating out"

- For continuous r.v.'s, $p(X) = \int_y p(X, Y = y)dy, \quad p(Y) = \int_x p(X = x, Y)dx$

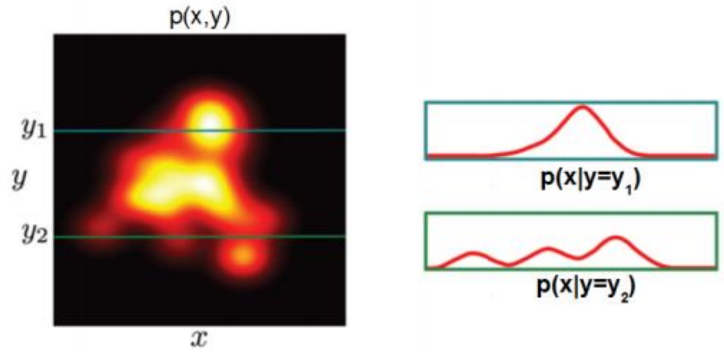
Conditional Probability Distribution

- Consider two r.v.'s X and Y (discrete/continuous – both need not of same type)
- Conditional PMF/PDF $p(X|Y)$ is the prob. dist. of one r.v. X, fixing other r.v. Y
- $p(X|Y = y)$ or $p(Y|X = x)$ like taking a slice of the joint dist. $p(X,Y)$

Discrete Random Variables



Continuous Random Variables



- Note: A conditional PMF/PDF may also be conditioned on something that is not the value of an r.v. but some fixed quantity in general

We will see cond. dist. of output $\boldsymbol{y}$ given weights $\boldsymbol{w}$ (r.v.) and features $\boldsymbol{X}$ written as $p(\boldsymbol{y}|\boldsymbol{w},\boldsymbol{X})$

Some Basic Rules

- **Sum Rule:** Gives the marginal probability distribution from joint probability distribution

For discrete r.v.: $p(X) = \sum_Y p(X, Y)$

For continuous r.v.: $p(X) = \int_Y p(X, Y) dY$

- **Product Rule:** $p(X, Y) = p(Y|X)p(X) = p(X|Y)p(Y)$

- **Bayes' rule:** Gives conditional probability distribution (can derive it from product rule)

$$\boxed{p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}}$$

For discrete r.v.: $p(Y|X) = \frac{p(X|Y)p(Y)}{\sum_Y p(X|Y)p(Y)}$

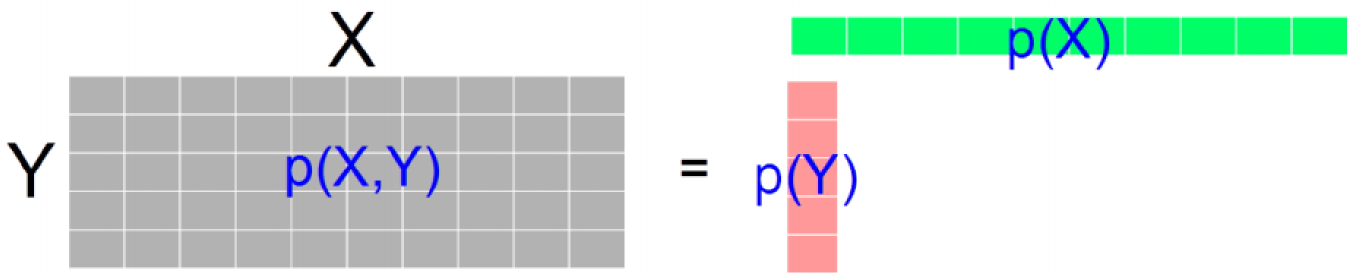
For continuous r.v.: $p(Y|X) = \frac{p(X|Y)p(Y)}{\int_Y p(X|Y)p(Y)dY}$

- **Chain Rule:** $p(X_1, X_2, \ldots, X_N) = p(X_1)p(X_2|X_1)\ldots p(X_N|X_1, \ldots, X_{N-1})$

Independence

- X and Y are independent when knowing one tells nothing about the other

$$
\begin{aligned}
p(X|Y=y) &= p(X)\\
p(Y|X=x) &= p(Y)\\
p(X,Y) &= p(X)p(Y)
\end{aligned}
$$



- The above is the marginal independence $(X \perp\!\!\!\perp Y)$

- Two r.v.'s X and Y may not be marginally indep but may be given the value of another r.v. Z

$$p(X,Y|Z=z) = p(X|Z=z)p(Y|Z=z) \qquad X \perp\!\!\!\perp Y|Z$$

Coming up next

- Some other basic concepts from probability and statistics
- Probabilistic models and parameter estimation in probabilistic models
  - MLE, MAP, Bayesian approaches

Probabilistic Machine Learning (3): Parameter Estimation via Maximum Likelihood

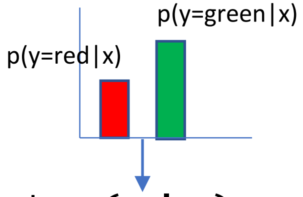
CS771: Introduction to Machine Learning

Piyush Rai

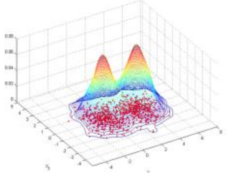
Probabilistic ML: Some Motivation

- In many ML problems, we want to model and reason about data probabilistically

- At a high-level, this is the density estimation view of ML, e.g.,



  - Given input-output pairs $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$ estimate the conditional $p(y|\boldsymbol{x})$
  - Given inputs $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, estimate the distribution $p(\boldsymbol{x})$ of the inputs



  - Note 1: These dist. will depend on some parameters $\boldsymbol{\theta}$ (to be estimated), and written as

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

  - Note 2: These dist. sometimes assumed to have a specific form, but sometimes not

- Assuming the form of the distribution to be known, the goal in estimation is to use the observed data to estimate the parameters of these distributions

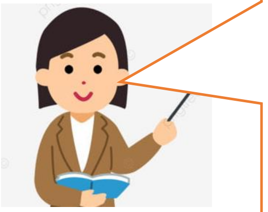
Probabilistic Modeling: The Basic Idea

- Assume N observations $\boldsymbol{y} = \{y_1, y_2, \dots, y_N\}$, generated from a presumed prob. model

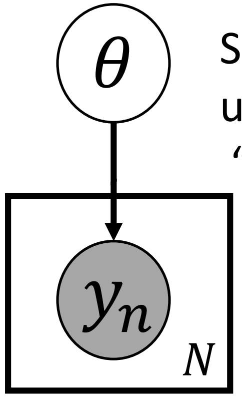
$$y_n \sim p(y|\theta) \qquad \forall n$$

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

- Here $p(y|\theta)$ is a conditional distribution, conditioned on params $\boldsymbol{\theta}$ (to be learned)
  - Note: $\boldsymbol{\theta}$ may be fixed unknown or an unknown random variable (we will study both cases)



The parameters $\boldsymbol{\theta}$ may themselves depend on other unknown/known parameters (called hyperparameters), which may depend on other unknowns, and so on. ☺ This is essentially "hierarchical" modeling (will see various examples later)



Such diagrams are usually called the "plate notation"

The Predictive dist. tells us how likely each possible value of a new observation $\boldsymbol{y}_*$ is. Example: if $\boldsymbol{y}_*$ denotes the outcome of a coin toss, then what is $p(\boldsymbol{y}_* = \text{"}\boldsymbol{head}\text{"}|\boldsymbol{y})$, given $\boldsymbol{N}$ previous coin tosses $\boldsymbol{y} = \{y_1, y_2, \dots, y_N\}$

- Some of the tasks that we may be interested in
  - Parameter estimation: Estimating the unknown parameters $\boldsymbol{\theta}$ (and other unknowns $\boldsymbol{\theta}$ depends on)
  - Prediction: Estimating the **predictive distribution** of new data, i.e., $p(\boldsymbol{y}_*|\boldsymbol{y})$ - this is also a conditional distribution (conditioned on past data $\boldsymbol{y} = \{y_1, y_2, \dots, y_N\}$, as well as $\boldsymbol{\theta}$ and other things)

Parameter Estimation in Probabilistic Models

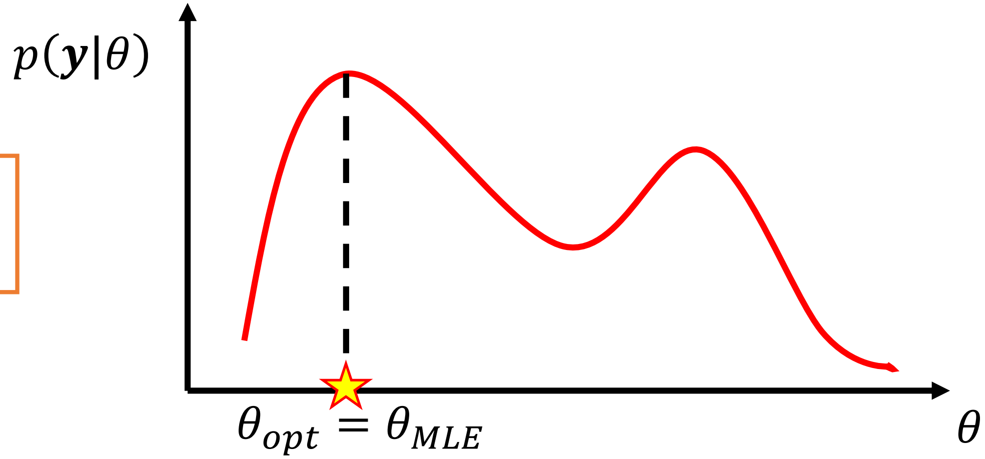
- Since data is assumed to be i.i.d., we can write down its total probability as

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

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

- $p(\boldsymbol{y}|\theta)$ called "likelihood" - probability of observed data as a function of params $\boldsymbol{\theta}$

How do I find the best $\boldsymbol{\theta}$?

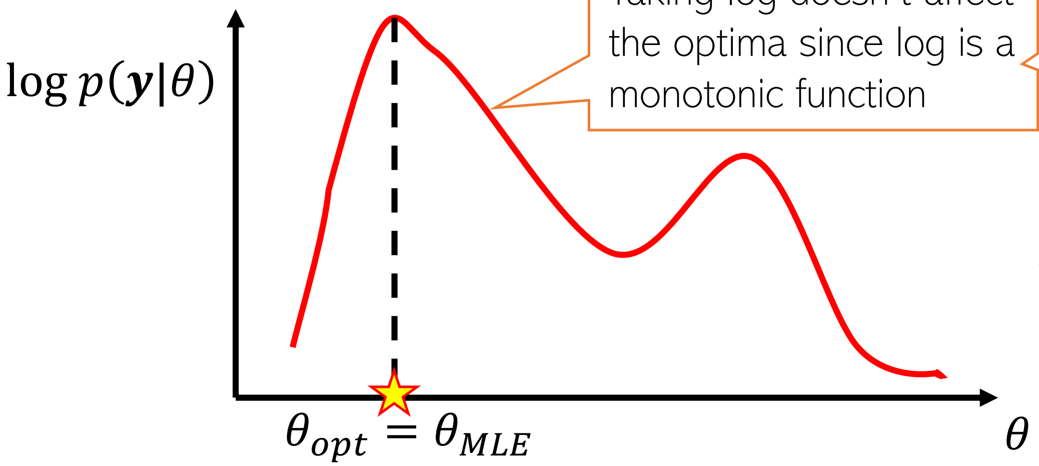


Well, one option is to find the $\boldsymbol{\theta}$ that maximizes the likelihood (probability of the observed data) – basically, which value of $\boldsymbol{\theta}$ makes the observed data most likely to have come from the assumed distribution $p(y|\theta)$ --- Maximum Likelihood Estimation (MLE)

- In parameter estimation, the goal is to find the "best" $\boldsymbol{\theta}$, given observed data $\boldsymbol{y}$
- Note: Instead of finding single best, sometimes may be more informative to learn a distribution for $\boldsymbol{\theta}$ (can tell us about uncertainty in our estimate of $\boldsymbol{\theta}$ – more later)

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 om 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). Note: $\boldsymbol{\theta}$ may have constraints

Maximum Likelihood Estimation (MLE)

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

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

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

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

- 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>

Indeed. It may overfit. Several ways to prevent it: Use regularizer or other strategies to prevent overfitting. Alternatives, use "prior" distributions on the parameters $\boldsymbol{\theta}$ 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

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: $\sum_{n=1}^{N} \log p(y_n|\theta) = \sum_{n=1}^{N} [y_n \log \theta + (1-y_n)\log(1-\theta)]$

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

- Maximizing log-lik (or minimizing NLL) w.r.t. θ will give a closed form expression

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

Coming up next

- Prior distributions and their role in parameter estimation
  - Maximum-a-Posteriori (MAP) Estimation
  - Fully Bayesian inference
- Probabilistic modeling for regression and classification problems

Probabilistic Machine Learning (2): Probability Basics (Contd)

CS771: Introduction to Machine Learning

Piyush Rai

Expectation

- Expectation of a random variable tells the expected or average value it takes

- Expectation of a discrete random variable $X \in S_X$ having PMF $p(X)$

$$\mathbb{E}[X] = \sum_{x \in S_X} x p(x)$$

Probability that $X = x$

- Expectation of a continuous random variable $X \in S_X$ having PDF $p(X)$

$$\mathbb{E}[X] = \int_{x \in S_X} x p(x) dx$$

Probability density at $X = x$

Note that this exp. is w.r.t. the distribution $p(f(X))$ of the r.v. $f(X)$

- The definition applies to functions of r.v. too (e.g.., $\mathbb{E}[f(X)]$)

Often the subscript is omitted but do keep in mind the underlying distribution

- Exp. is always w.r.t. the prob. dist. $p(X)$ of the r.v. and often written as $\mathbb{E}_p[X]$

Expectation: A Few Rules

X and Y need not be even independent. Can be discrete or continuous

- Expectation of sum of two r.v.'s: $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$
- Proof is as follows
  - Define $Z = X + Y$

$$\mathbb{E}[Z] = \textstyle\sum_{z\in S_Z} z\cdot p(Z=z) \qquad \text{s.t. } z = x+y \text{ where } x \in S_X \text{ and } y \in S_Y$$

$$= \textstyle\sum_{x\in S_X}\sum_{y\in S_Y}(x+y)\cdot p(X=x,Y=y)$$

$$= \textstyle\sum_x\sum_y x\cdot p(X=x,Y=y) + \sum_x\sum_y y\cdot p(X=x,Y=y)$$

$$= \textstyle\sum_x x\sum_y p(X=x,Y=y) + \sum_y y\sum_x p(X=x,Y=y)$$

$$= \textstyle\sum_x x\cdot p(X=x) + \sum_y y\cdot p(Y=y)$$

Used the rule of marginalization of joint dist. of two r.v.'s

$$= \mathbb{E}[X] + \mathbb{E}[Y]$$

Expectation: A Few Rules (Contd)

α is a real-valued scalar

α and β are real-valued scalars

f and g are arbitrary functions.

- Expectation of a scaled r.v.: $\mathbb{E}[\alpha X] = \alpha\mathbb{E}[X]$
- Linearity of expectation: $\mathbb{E}[\alpha X + \beta Y] = \alpha\mathbb{E}[X] + \beta\mathbb{E}[Y]$
- (More General) Lin. of exp.: $\mathbb{E}[\alpha f(X) + \beta g(Y)] = \alpha\mathbb{E}[f(X)] + \beta\mathbb{E}[g(Y)]$
- Exp. of product of two independent r.v.'s: $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$
- Law of the Unconscious Statistician (LOTUS): Given an r.v. X with a known prob. dist. $p(X)$ and another random variable $Y = g(X)$ for some function g

Requires finding $p(Y)$

Requires only $p(X)$ which we already have

$$\mathbb{E}[Y] = \mathbb{E}[g(X)] = \sum_{y \in S_Y} y p(y) \quad = \sum_{x \in S_X} g(x) p(x)$$

LOTUS also applicable for continuous r.v.'s

- Rule of iterated expectation: $\mathbb{E}_{p(X)}[X] = \mathbb{E}_{p(Y)}[\mathbb{E}_{p(X|Y)}[X|Y]]$

Variance and Covariance

- Variance of a scalar r.v. tells us about its spread around its mean value $\mathbb{E}[X] = \mu$

$$\textbf{var}[X] = \mathbb{E}[(X-\mu)^2] = \mathbb{E}[X^2] - \mu^2$$

- Standard deviation is simply the square root is variance
- For two scalar r.v.'s X and Y, the covariance is defined by

$$\textbf{cov}[X,Y] = \mathbb{E}[\{X-\mathbb{E}[X]\}\{Y-\mathbb{E}[Y]\}] = \mathbb{E}[XY] - \mathbb{E}[\mathrm{X}]\mathbb{E}[\mathrm{Y}]$$

- For two vector r.v.'s $\boldsymbol{X}$ and $\boldsymbol{Y}$ (assume column vec), the covariance matrix is defined by

$$\textbf{cov}[\boldsymbol{X},\boldsymbol{Y}] = \mathbb{E}[\{\boldsymbol{X}-\mathbb{E}[\boldsymbol{X}]\}\{\boldsymbol{Y}^\top-\mathbb{E}[\boldsymbol{Y}^\top]\}] = \mathbb{E}[\boldsymbol{X}\boldsymbol{Y}^\top] - \mathbb{E}[\mathbf{X}]\mathbb{E}[\boldsymbol{Y}^\top]$$

- Cov. of components of a vector r.v. $\boldsymbol{X}$: $\textbf{cov}[\boldsymbol{X}] = \textbf{cov}[\boldsymbol{X},\boldsymbol{X}]$
- Note: The definitions apply to functions of r.v. too (e.g., $\textbf{var}[f(X)]$)
- Note: Variance of sum of independent r.v.'s: $\textbf{var}[X + Y] = \textbf{var}[X] + \textbf{var}[Y]$

Important result

Transformation of Random Variables

- Suppose $Y = f(X) = AX + b$ be a linear function of a vector-valued r.v. X (A is a matrix and b is a vector, both constants)
- Suppose $\mathbb{E}[X] = \mu$ and $\mathbf{cov}[X] = \Sigma$, then for the vector-valued r.v. Y

$$\mathbb{E}[Y] = \mathbb{E}[AX + b] = A\mu + b$$

$$\text{cov}[Y] = \text{cov}[AX + b] = A\Sigma A^\top$$

- Likewise, if $Y = f(X) = a^\top X + b$ be a linear function of a vector-valued r.v. X (a is a vector and b is a scalar, both constants)
- Suppose $\mathbb{E}[X] = \mu$ and $\mathbf{cov}[X] = \Sigma$, then for the scalar-valued r.v. Y

$$\mathbb{E}[Y] = \mathbb{E}[a^\top X + b] = a^\top \mu + b$$

$$\text{var}[Y] = \text{var}[a^\top X + b] = a^\top \Sigma a$$

Common Probability Distributions

Important: We will use these extensively to model data as well as parameters of models

- Some common discrete distributions and what they can model
  - **Bernoulli:** Binary numbers, e.g., outcome (head/tail, 0/1) of a coin toss
  - **Binomial:** Bounded non-negative integers, e.g., # of heads in n coin tosses
  - **Multinomial/multinoulli:** One of K (>2) possibilities, e.g., outcome of a dice roll
  - **Poisson:** Non-negative integers, e.g., # of words in a document
- Some common continuous distributions and what they can model
  - **Uniform:** numbers defined over a fixed range
  - **Beta:** numbers between 0 and 1, e.g., probability of head for a biased coin
  - **Gamma:** Positive unbounded real numbers
  - **Dirichlet:** vectors that sum of 1 (fraction of data points in different clusters)
  - **Gaussian:** real-valued numbers or real-valued vectors

Coming up next

- Probabilistic Modeling
- Basics of parameter estimation for probabilistic models

Probabilistic Machine Learning (4): Parameter Estimation: MAP and Bayesian Inference

CS771: Introduction to Machine Learning

Piyush Rai

MLE and Its Shortcomings..

- MLE finds parameters that make the observed data most probable

Log-likelihood

Neg. log-likelihood (NLL)

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

- 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 probability distribution over the learned params instead of point est

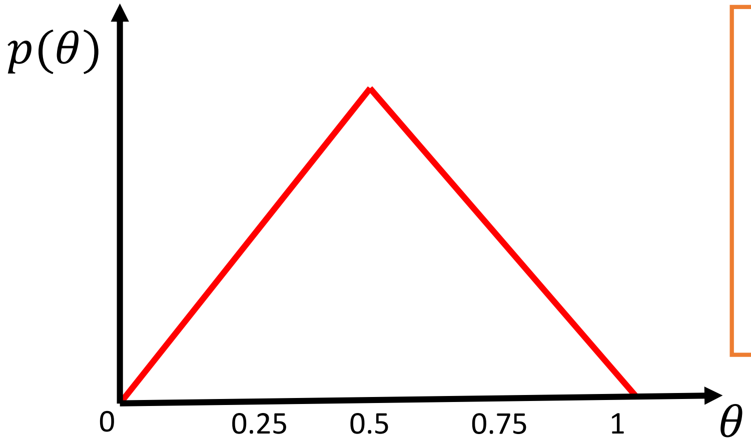
This distribution can give us a sense about the uncertainty in the parameter estimate

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

Priors

- 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 $\boldsymbol{\theta}$ 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

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

Posterior

Prior

Likelihood

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 – CS775 does that in more detail)

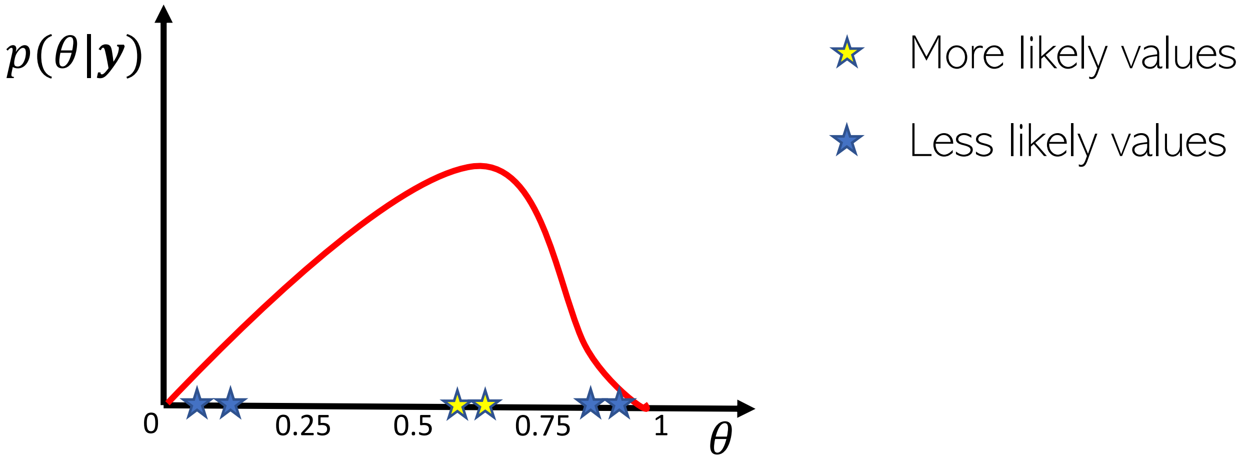
- Two way now to report the answer now:
  - Report the maxima (mode) of the posterior: $\arg\max_{\theta} p(\theta|\boldsymbol{y})$
  - Report the full posterior (and its properties, e.g., mean, mode, variance, quantiles, etc)

Maximum-a-posteriori (MAP) estimation

Fully Bayesian inference

Posterior

- Posterior distribution tells us how probable different parameter values are after 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. $\boldsymbol{\theta}$, 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. ☺

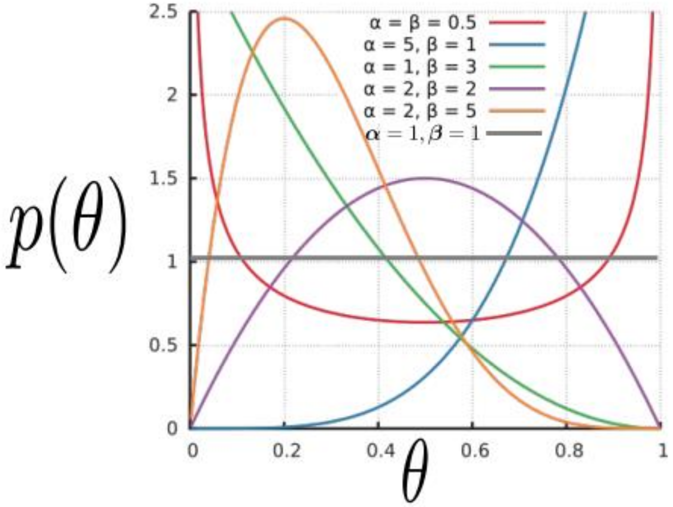
- Same as MLE with an extra log-prior-distribution term (acts as a regularizer) ☺
- If the prior is absent or uniform (all values equally likely a prior) then MAP=MLE

MAP Estimation: An Example

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

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

- Also need a prior since we want to do MAP estimation
- Since $\theta \in (0,1)$, a reasonable choice of prior for $\boldsymbol{\theta}$ 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

$\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (both non-negative reals) are the two hyperparameters of this Beta prior

Using $\boldsymbol{\alpha} = \mathbf{1}$ and $\boldsymbol{\beta} = \mathbf{1}$ will make the Beta prior a uniform 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 $\boldsymbol{\alpha} = \mathbf{1}$ and $\boldsymbol{\beta} = \mathbf{1}$ gives us the same solution as MLE

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

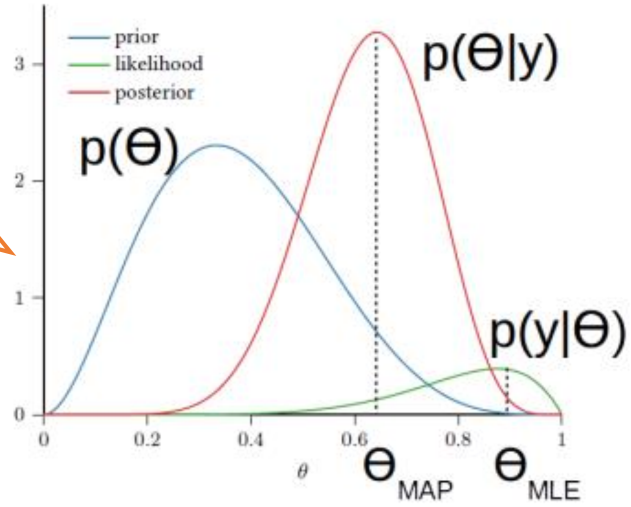
Prior's hyperparameters have an interesting interpretation. Can think of $\alpha - 1$ and $\beta - 1$ as the number of heads and tails, respectively, before starting the coin-toss experiment (akin to "pseudo-observations")

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

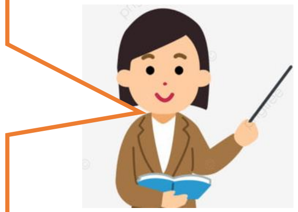
Fully Bayesian Inference

- 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

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

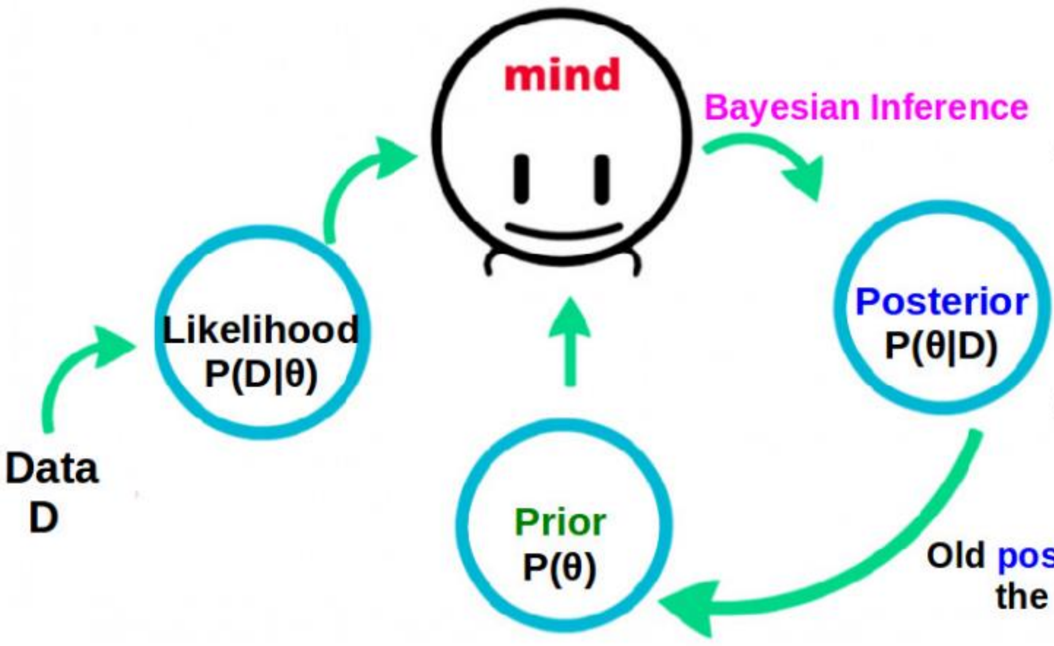
Computable analytically only when the prior likelihood are "friends" with each other (i.e., they form a conjugate pair of distributions (distributions from exponential family have conjugate priors

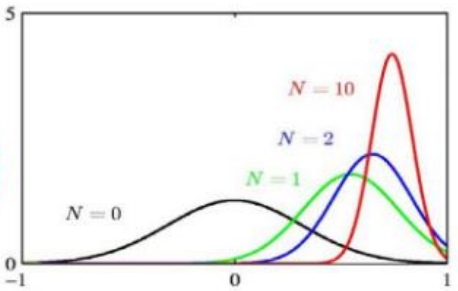
An example: Bernoulli and Beta are conjugate. Will see some more such pairs

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

- 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

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)

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 $\theta : p(\mathbf{y}) = \int p(\theta,\mathbf{y})d\theta = \int p(\theta)p(\mathbf{y}|\theta)d\theta$

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 ..

Not true in general, but in some cases (e.g., when mean of the Gaussian prior is fixed)

- 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 $\boldsymbol{\theta}$

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 $\boldsymbol{\theta}$, $\boldsymbol{y}_*$ does not depend on $\boldsymbol{y}$

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

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

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

- When doing fully Bayesian est, getting the predictive dist. Will require computing

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

This computes the predictive distribution by averaging over the full posterior – basically calculate $p(y_*|\theta)$ for each possible $\boldsymbol{\theta}$, weighs it by how likely this $\boldsymbol{\theta}$ is under the posterior $\boldsymbol{p(\theta|y)}$, 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

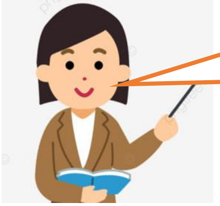
$$\theta_{MLE} = \frac{N_1}{N} \qquad \theta_{MAP} = \frac{N_1+\alpha-1}{N+\alpha+\beta-2} \qquad p(\theta|\boldsymbol{y}) = \text{Beta}(\theta|\alpha+N_1, \beta+N_0)$$

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

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

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

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



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 $\boldsymbol{\theta}$ 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)

Coming up next

- Probabilistic modeling for regression and classification problems

Probabilistic Models for Supervised Learning(1): Probabilistic Linear Regression

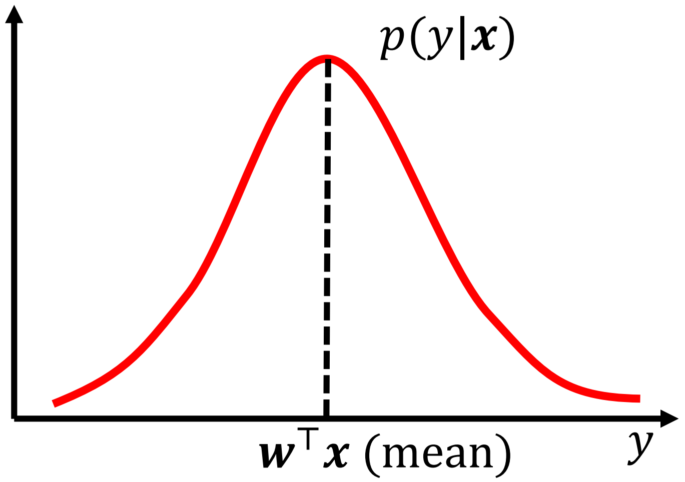
CS771: Introduction to Machine Learning

Piyush Rai

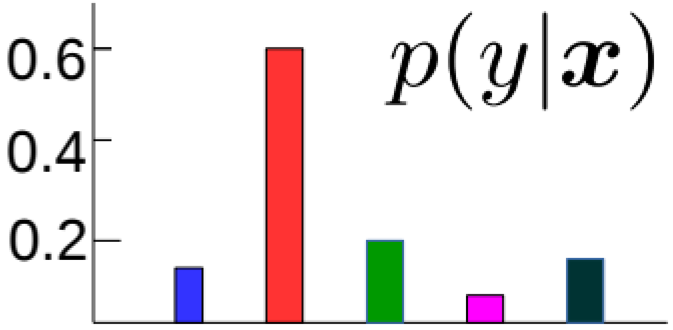
Probabilistic Models for Supervised Learning

- Goal: Learn the conditional distribution of output given input, i.e., $\boldsymbol{p(y|x)}$

Probabilistic Linear Regression



Probabilistic Classification



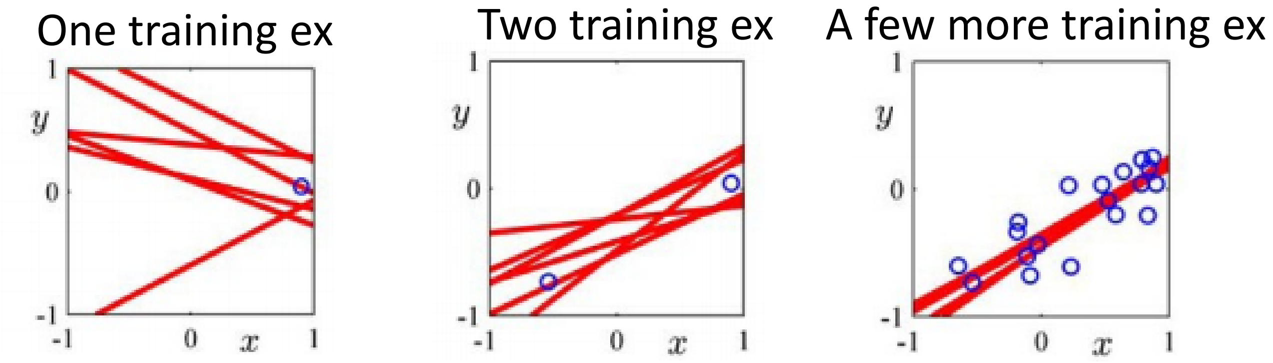
- $\boldsymbol{p(y|x)}$ is more informative than a single prediction $\boldsymbol{y}$
  - From $\boldsymbol{p(y|x)}$, can get "expected" or "most likely" output $\boldsymbol{y}$
  - For classifn, "soft" predictions (e.g., rather than yes/no, prob. of "yes")
  - "Uncertainty" in the predicted output $\boldsymbol{y}$ (e.g., by looking at the variance of $\boldsymbol{p(y|x)}$)

Such uncertainty also helps in "active learning" where we wish to identify "difficult" (and hence more useful) training examples

- Can also learn a distribution over the model params using fully Bayesian inference

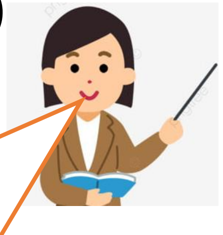
Distribution over model parameters??

- Recall that linear/ridge regression gave a single “optimal” weight vector
- With a probabilistic model for linear regression, we have two options
  - Use MLE/MAP to get a single “optimal” weight vector
  - Use fully Bayesian inference to learn a distribution over weight vectors (figure below)


One training ex

Two training ex

A few more training ex



Rather than returning just a single “best” solution (a line in this example), the fully Bayesian approach would give us several “probable” lines (consistent with training data) by learning the full posterior distribution over the model parameters (each of which corresponds to a line)

$$p(y_*|\boldsymbol{X},\boldsymbol{y}) = \int p(y_*|\boldsymbol{w},\boldsymbol{x})\; p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y})d\boldsymbol{w}$$

Posterior predictive distribution by doing posterior weighted averaging over all possible $\boldsymbol{w}$, not just the most likely one. Thus more robust predictions especially if we are uncertain about the best solution.

Predictive distribution using a single $\boldsymbol{w}$ (plug-in predictive distribution)

How important/like this $\boldsymbol{w}$ is under the posterior distribution (its posterior probability)

In this course, we will mostly focus on probabilistic ML when using MLE/MAP and predictive distributions computed using a single best estimate (MLE/MAP). We will only briefly look some simple examples with fully Bayesian approach (CS772/775 covers this approach in greater depth)

Probabilistic Models for Supervised Learning

- Usually two ways to model the conditional distribution $\boldsymbol{p(y|x)}$
- Approach 1: Don't model $\boldsymbol{x}$, and model $\boldsymbol{p(y|x)}$ directly using a prob. distribution

"discriminative" sup learning

Gaussian distribution

$$p(y|\boldsymbol{x},\boldsymbol{w}) = \mathcal{N}(y|\boldsymbol{w}^\top\boldsymbol{x},\beta^{-1})$$

Probabilistic linear regression

The "sigmoid" function

$$p(y|\boldsymbol{x},\boldsymbol{w}) = \text{Bernoulli}(y|\sigma(\boldsymbol{w}^\top\boldsymbol{x}))$$

Probabilistic linear binary classification

We assume the conditional distribution to be some appropriate distribution and treat the weights $\boldsymbol{w}$ as learnable parameters of the model (using MLE/MAP/fully Bayesian inference). Need not be a linear model – can replace $\boldsymbol{w}^\top\boldsymbol{x}$ by a nonlinear function $f(\boldsymbol{x})$

- Approach 2: Model both $\boldsymbol{x}$ and $\boldsymbol{y}$ via their joint distr. and get the conditional as

"generative" sup learning

Called "generative" because we are learning the generative distributions for output as well as inputs

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

Here $\boldsymbol{\theta}$ denotes all the model parameters that we need to model the joint distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$ (will see examples later)

Prob. distribution of inputs from class k

$$p(y=k|\boldsymbol{x},\theta) = \frac{p(\boldsymbol{x},y=k|\theta)}{p(\boldsymbol{x}|\theta)} = \frac{p(\boldsymbol{x}|y=k,\theta)p(y=k|\theta)}{\sum_{\ell=1}^{K} p(\boldsymbol{x}|y=\ell,\theta)p(y=\ell|\theta)}$$

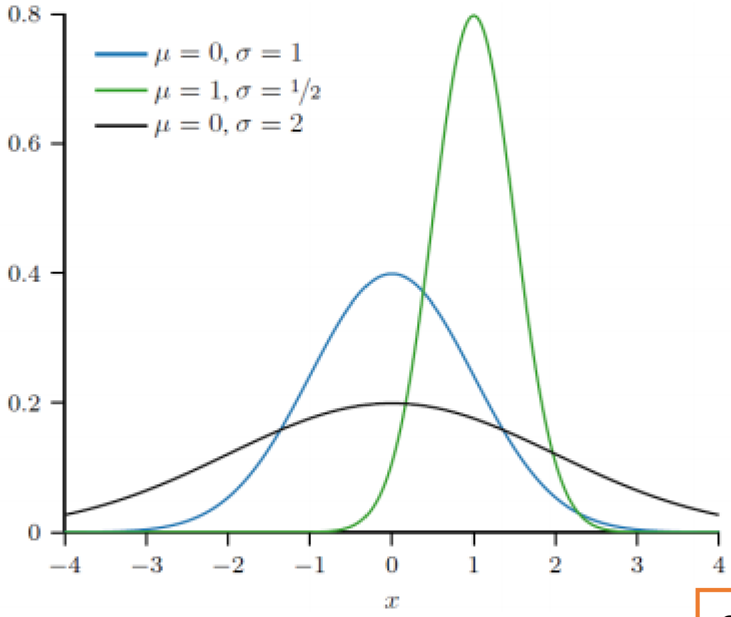
For a multi-class classification model with K classes

Brief Detour (Gaussian Distribution)

Gaussian Distribution (Univariate)

- Distribution over real-valued scalar random variables $x \in \mathbb{R}$
- Defined by a scalar mean μ and a scalar variance σ^2

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$



- Mean: $\mathbb{E}[x] = \mu$
- Variance: $\text{var}[x] = \sigma^2$
- Inverse of variance is called precision: $\beta = \frac{1}{\sigma^2}$.

Gaussian PDF in terms of precision

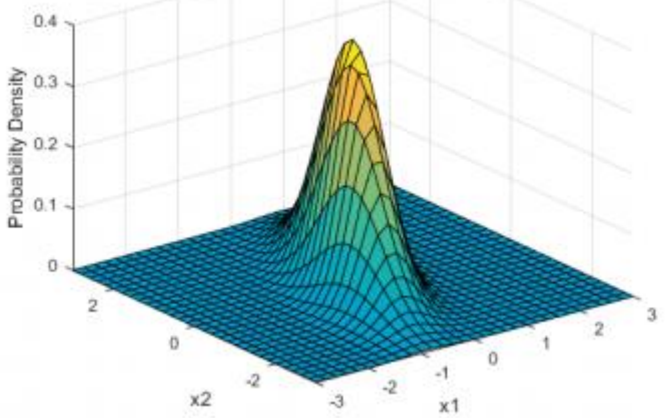
$$\mathcal{N}(x|\mu,\beta) = \sqrt{\frac{\beta}{2\pi}} \exp\left[-\frac{\beta}{2}(x-\mu)^2\right]$$

Gaussian Distribution (Multivariate)

- Distribution over real-valued vector random variables $\boldsymbol{x} \in \mathbb{R}^D$
- Defined by a mean vector $\boldsymbol{\mu} \in \mathbb{R}^D$ and a covariance matrix $\boldsymbol{\Sigma}$

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}|}}\exp[-(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})]$$

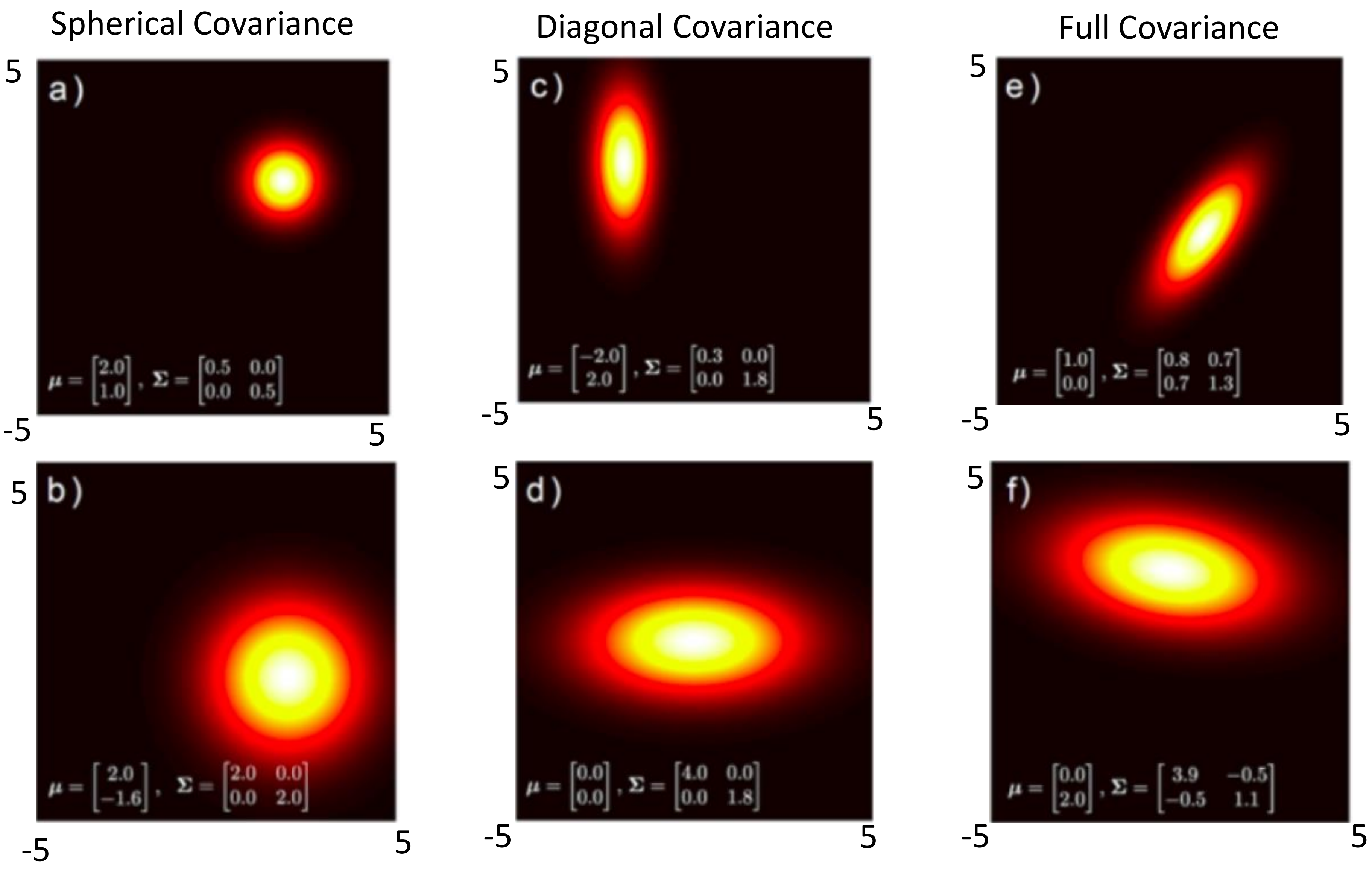
A two-dimensional Gaussian



- Note: The cov. matrix $\boldsymbol{\Sigma}$ must be symmetric and PSD
  - All eigenvalues are positive
  - $\boldsymbol{z}^\top\boldsymbol{\Sigma}\boldsymbol{z} \geq \boldsymbol{0}$ for any real vector $\boldsymbol{z}$

- The covariance matrix also controls the shape of the Gaussian

Covariance Matrix for Multivariate Gaussian



Spherical: Equal spreads (variances) along all dimensions

Diagonal: Unequal spreads (variances) along all directions but still axis-parallel

Full: Unequal spreads (variances) along all directions and also spreads along oblique directions

Probabilistic Linear Regression

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1})$$

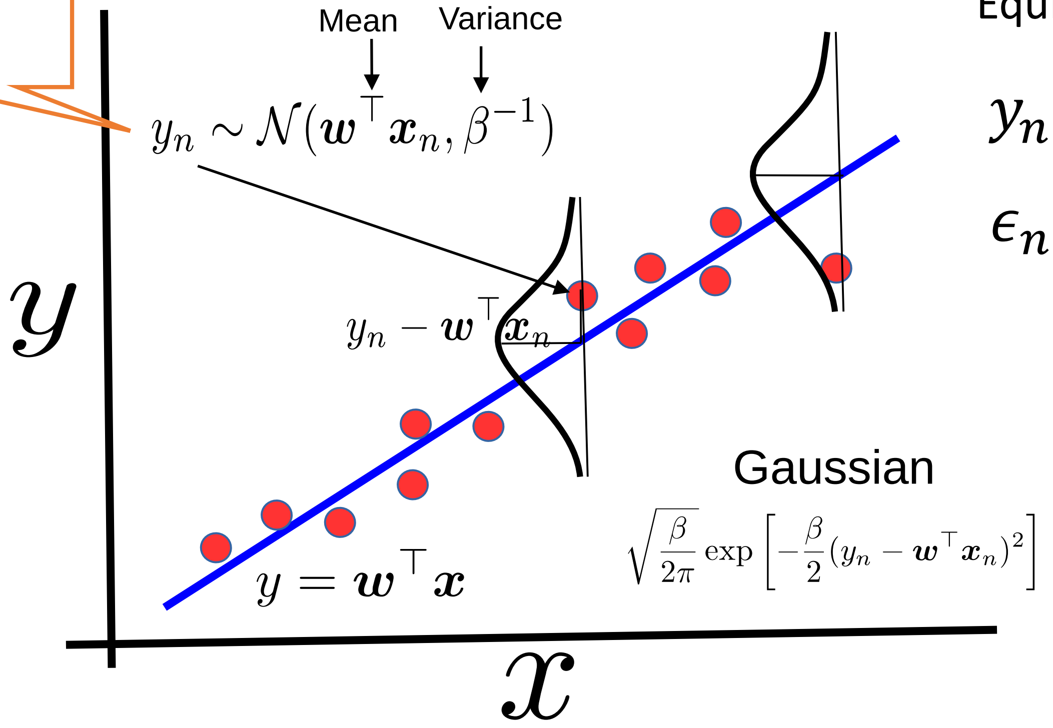
Gaussian distribution

Other distributions can also be used for probabilistic linear regression (e.g., Laplace) as we will see later

Linear Regression: A Probabilistic View

Defines our likelihood model: $p(y_n|\boldsymbol{w}, \boldsymbol{x}_n)$ - Gaussian

Output y_n assumed generated from a Gaussian with mean $\boldsymbol{w}^\top \boldsymbol{x}_n$



Output y_n generated from a linear model and then zero mean Gaussian noise added

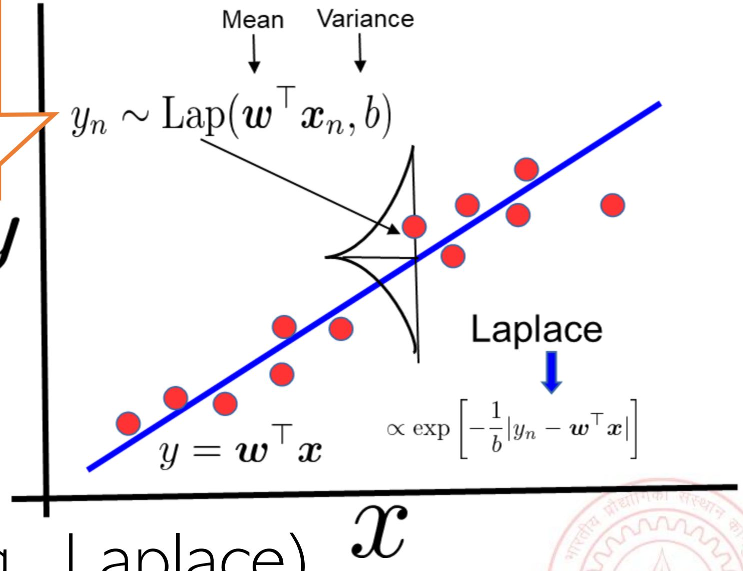
Equivalently:

$$y_n = \boldsymbol{w}^\top \boldsymbol{x}_n + \epsilon_n$$

$$\epsilon_n \sim \mathcal{N}(0, \beta^{-1})$$

Note the term in the Gaussian's exponent – just like a squared error we saw for least squares regression ☺

Using a Laplace distribution would correspond to using an absolute loss



- Several variants of this basic model are possible
  - Other distributions to model the additive noise (e.g., Laplace)
  - Different noise variance/precision for each output: $y_n \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \beta_n^{-1})$

Heteroskedastic noise

MLE for Probabilistic Linear Regression

- Since each likelihood term is a Gaussian, we have

Also note that $\boldsymbol{x}_n$ is fixed here but the likelihood depend on it, so it is being conditioned on

Omitting β from the conditioning side for brevity

$$p(y_n|\boldsymbol{w},\boldsymbol{x}_n) = \mathcal{N}(y_n|\boldsymbol{w}^\top\boldsymbol{x}_n,\beta^{-1}) = \sqrt{\frac{\beta}{2\pi}}\exp\left[-\frac{\beta}{2}(y_n-\boldsymbol{w}^\top\boldsymbol{x}_n)^2\right]$$

Exercise: Verify that you can also write the overall likelihood as a single $\boldsymbol{N}$ dimensional Gaussian with mean $\boldsymbol{Xw}$ and cov. matrix $\beta^{-1}\boldsymbol{I}_N$

- Thus the overall likelihood (assuming i.i.d. responses) will be

$$p(\boldsymbol{y}|\mathbf{X},\boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n,\boldsymbol{w}) = \left(\frac{\beta}{2\pi}\right)^{N/2}\exp\left[-\frac{\beta}{2}\sum_{n=1}^{N}(y_n-\boldsymbol{w}^\top\mathbf{x}_n)^2\right]$$

- Log-likelihood (ignoring constants w.r.t. $\boldsymbol{w}$)

$$\boxed{\log p(\boldsymbol{y}|\mathbf{X},\boldsymbol{w}) \propto -\frac{\beta}{2}\sum_{n=1}^{N}(y_n-\boldsymbol{w}^\top\mathbf{x}_n)^2}$$

MLE for probabilistic linear regression with Gaussian noise is equivalent to least squares regression without any regularization (with solution $\hat{w}_{MLE} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$

- Negative log likelihood (NLL) in this case is similar to squared loss function

MAP Estimation for Prob. Lin. Reg.: The Prior

- For MAP estimation, we need a prior distribution over the parameters $\boldsymbol{w} \in \mathbb{R}^D$
- A reasonable prior for real-valued vectors can be a multivariate Gaussian

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_0, \boldsymbol{\Sigma})$$

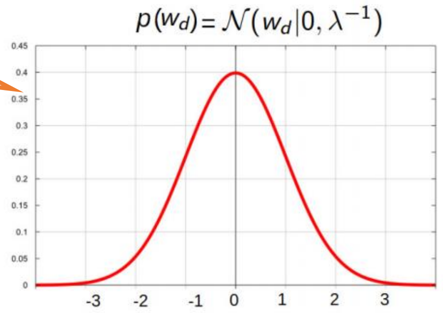
Equivalent to saying that *a priori* we expect the solution to be close to some vector $\boldsymbol{w}_0$
(subject to $\boldsymbol{\Sigma}$ being such that the variances is not too large

- A specific example of a multivariate Gaussian prior in this problem

Omitting
λ for brevity

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \lambda^{-1}\boldsymbol{I}_D) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1}) = \prod_{d=1}^{D} p(w_d)$$

The precision λ of the Gaussian prior controls how aggressively the prior pushes the elements towards mean (0)



This is essentially like a regularizer that pushes elements of $\boldsymbol{w}$ to be small (we will see shortly)

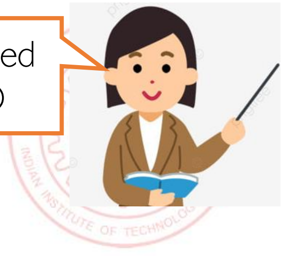
Equivalent to saying that *a priori* we expect each element of the solution to be close to 0 (i.e., “small”)

$$\mathcal{N}(w_d|0, \lambda^{-1}) = \sqrt{\frac{\lambda}{2\pi}} \exp\left[-\frac{\lambda}{2} w_d^2\right]$$

$$\mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \lambda^{-1}\boldsymbol{I}_D) = \left(\frac{\lambda}{2\pi}\right)^{D/2} \exp\left[-\frac{\lambda}{2}\sum_{d=1}^{D} w_d^2\right] = \left(\frac{\lambda}{2\pi}\right)^{D/2} \exp\left[-\frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w}\right]$$

Aha! This $\boldsymbol{w}^\top\boldsymbol{w}$ term reminds me of the ℓ_2 regularizer ☺

That’s indeed the case ☺



MAP Estimation for Probabilistic Linear Regression

- The MAP objective (log-posterior) will be the log-likelihood + $\log p(\boldsymbol{w})$

$$-\frac{\beta}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 - \frac{\lambda}{2}\boldsymbol{w}^\top \boldsymbol{w}$$

In the likelihood and prior, ignored terms that don't depend on $\boldsymbol{w}$

- Maximizing this is equivalent to minimizing the following w.r.t. $\boldsymbol{w}$

$$\boxed{\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\lambda}{\beta}\boldsymbol{w}^\top \boldsymbol{w}}$$

Not surprising since MAP estimation indeed optimizes a regularized loss function! ☺

- This is equivalent to ridge regression with regularization hyperparameter $\frac{\lambda}{\beta}$
- The solution will be $\widehat{\boldsymbol{w}}_{MAP} = (\boldsymbol{X}^\top \boldsymbol{X} + \frac{\lambda}{\beta}\boldsymbol{I}_D)^{-1}\boldsymbol{X}^\top \boldsymbol{y}$

Fully Bayesian Inference for Prob. Linear Regression

- Can also compute the full posterior distribution over $\boldsymbol{w}$

$$p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\boldsymbol{w})p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})}$$

For brevity, we have not shown the dependence of the various distributions here on the hyperparameters λ and β

- Likelihood and prior are conjugate (both Gaussians) - posterior will be Gaussian

$$p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$

$$\boldsymbol{\mu}_N = (\boldsymbol{X}^\top\boldsymbol{X} + \frac{\lambda}{\beta}\boldsymbol{I}_D)^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

$$\boldsymbol{\Sigma}_N = (\beta\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}_D)^{-1}$$

Posterior's mean is the same as the MAP solution since the mean and mode of a Gaussian are the same!

Note: λ and β are assumed to be fixed; otherwise, the problem is a bit harder (beyond the scope of CS771)

Deriving this result requires a bit of algebra (not too hard though).

We already know that the result will be Gaussian (due to conjugacy) – just need to multiply and rearrange terms to bring the result into a Gaussian form and identify the mean and covariance of that Gaussian – can be done using the "completing the squares" trick. Don't even need to worry about calculating the marginal. Will provide a note

Alternatively, just think of the posterior $p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y})$ as a reverse conditional of the likelihood $p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w})$ and apply standard results of Gaussians distributions (see maths refresher slides from Week 0)

We now have a distribution over the possible solutions – it has a mean but we can generate other plausible solutions by sampling from this posterior. Each sample will give a weight vector

Prob. Linear Regression: The Predictive Distribution

- Want the predictive distribution $p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})$ of the output y_* for a new input $\boldsymbol{x}_*$
- With MLE/MAP estimate of $\boldsymbol{w}$, we will use the plug-in predictive

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) \approx p(y_*|\mathbf{x}_*, \mathbf{w}_{MLE}) \quad = \quad \mathcal{N}(\mathbf{w}_{MLE}^\top \mathbf{x}_*, \beta^{-1}) \qquad \text{- MLE prediction}$$

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) \approx p(y_*|\mathbf{x}_*, \mathbf{w}_{MAP}) \quad = \quad \mathcal{N}(\mathbf{w}_{MAP}^\top \mathbf{x}_*, \beta^{-1}) \qquad \text{- MAP prediction}$$

- When doing fully Bayesian inference, can compute the posterior predictive dist.

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int p(y_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}) d\mathbf{w}$$

Not true in general for Prob. Lin. Reg. but because the hyperparameters λ and β are treated as fixed

- Requires an integral but has a closed form

Mean prediction

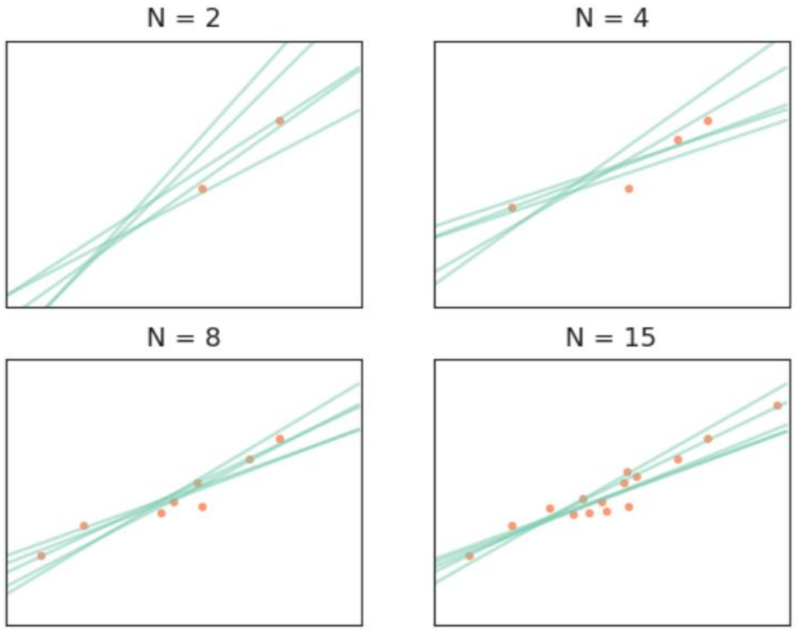
$$\boxed{p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_N^\top \mathbf{x}_*, \beta^{-1} + \mathbf{x}_*^\top \boldsymbol{\Sigma}_N \mathbf{x}_*)}$$

Input-specific predictive variance unlike the MLE/MAP based predictive where it was β^{-1} (and was same for all test inputs)

- Input-specific predictive uncertainty useful in problems where we want confidence estimates of the predictions made by the model (e.g., Active Learning)

Fully Bayesian Linear Regression – Pictorially

- Each sample from posterior $p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$ will give a weight vector $\boldsymbol{w}$
  - In case of lin. reg., each weight vector corresponds to a regression line



The posterior sort of represents an ensemble of solutions (not all are equally good but we can use all of them in an "importance-weighted" fashion to make the prediction using the posterior predictive distribution)

Importance of each solution in this ensemble is its posterior probability $p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y})$

- Each weight vector will give a different set of predictions on test data
  - These different predictions will give us a variance (uncertainty) estimate in model's prediction
  - The uncertainty decreases as $\boldsymbol{N}$ increases (we become more sure when we see more training data)

Pic source: https://waterprogramming.wordpress.com/

MLE, MAP/Fully Bayesian Lin. Reg: Summary

- MLE/MAP give point estimate of $\boldsymbol{w}$
  - MLE/MAP based prediction uses that single point estimate of $\boldsymbol{w}$
- Fully Bayesian approach gives the full posterior of $\boldsymbol{w}$
  - Fully Bayesian prediction does posterior averaging (computes posterior predictive distribution)
- Some things to keep in mind:
  - MLE estimation of a parameter leads to unregularized solutions
  - MAP estimation of a parameter leads to regularized solutions
  - A Gaussian likelihood model corresponds to using squared loss
  - A Gaussian prior on parameters acts as an ℓ_2 regularizer
  - Other likelihoods/priors can be chosen (result in other loss functions and regularizers)

E.g., using Laplace distribution for likelihood is equivalent to absolute loss, using it as a prior is equivalent to ℓ_1 regularization

- Can extend Bayesian linear regression to handle nonlinear regression
  - Using kernel based feature mapping $\phi(x)$: Gaussian Process regression

Evaluation Measures for Regression Models

- Plotting the prediction $\hat{y}_n$ vs truth y_n for the validation/test set
- Residual Sum of Squares (RSS) on the validation/test set

$$RSS(\mathbf{w}) = \sum_{n=1}^{N} (y_n - \hat{y}_n)^2$$

- RMSE (Root Mean Squared Error) $\triangleq \sqrt{\frac{1}{N} RSS(\mathbf{w})}$
- Coefficient of determination or R^2

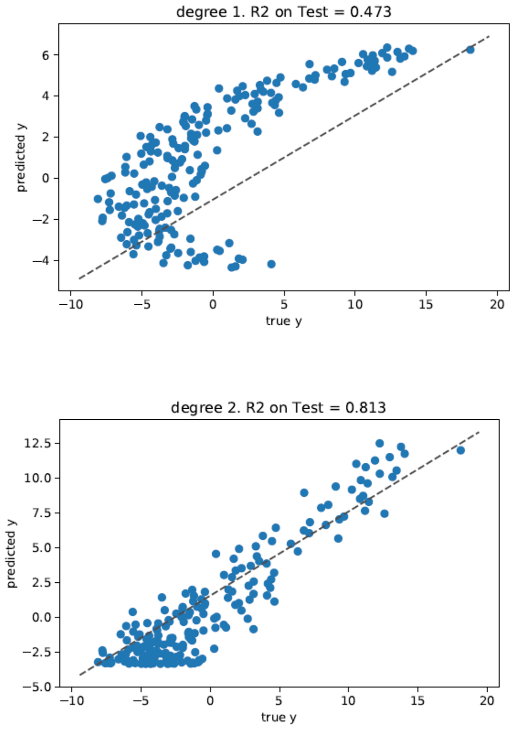
$$R^2 = 1 - \frac{\sum_{n=1}^{N}(y_n - \hat{y}_n)^2}{\sum_{n=1}^{N}(y_n - \bar{y})^2}$$

"relative" error w.r.t. a model that makes a constant prediction $\bar{y}$ for all inputs

Unlike RSS and RMSE, it is always between 0 and 1 and hence interpretable

$\bar{y}$ is empirical mean of true responses, i.e., $\frac{1}{N}\sum_{n=1}^{N} y_n$

Plots of true vs predicted outputs and R^2 for two regression models



Pic from MLAPP (Murphy)

Coming up next

- Probabilistic modeling classification problems
  - Logistic regression and softmax regression

Probabilistic Models for Supervised Learning(2): Logistic and Softmax Regression

CS771: Introduction to Machine Learning

Piyush Rai

Logistic Regression (LR)

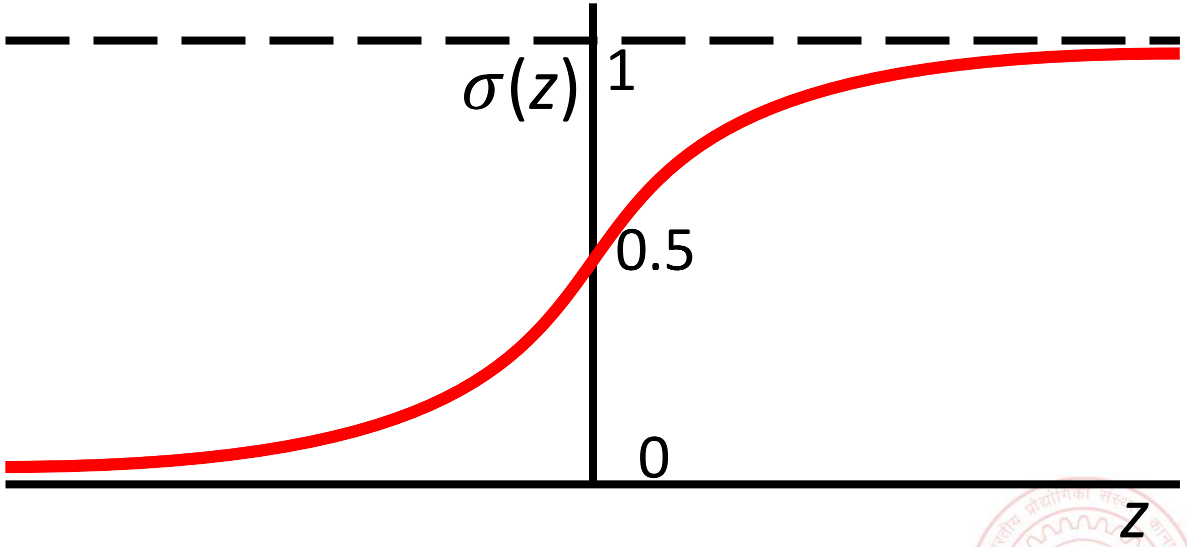
Multi-class extension known as "softmax regression"

Both very widely used

The word "regression" is a misnomer. Both are classification models

- A probabilistic model for binary classification
- Learns the PMF of the output label given the input, i.e., $p(y|\boldsymbol{x})$
- A discriminative model: Does not model inputs $\boldsymbol{x}$ (only relationship b/w $\boldsymbol{x}$ and y)
- Uses the sigmoid function to define the conditional probability of y being 1

$$\mu_x = p(y=1|\boldsymbol{w},\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top\boldsymbol{x})$$
$$= \frac{1}{1+\exp(-\boldsymbol{w}^\top\boldsymbol{x})}$$
$$= \frac{\exp(\boldsymbol{w}^\top\boldsymbol{x})}{1+\exp(\boldsymbol{w}^\top\boldsymbol{x})}$$



A linear model

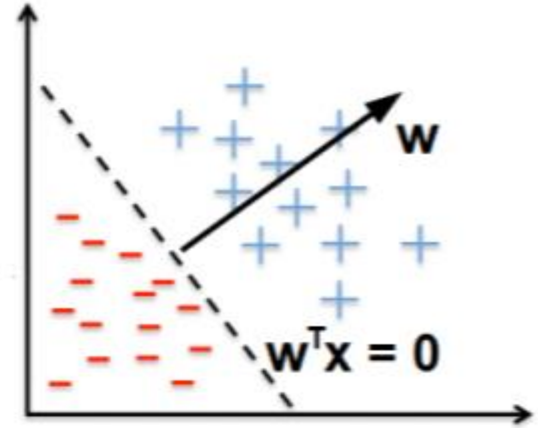
- Here $\boldsymbol{w}^\top\boldsymbol{x}$ is the score for input $\boldsymbol{x}$. The sigmoid turns it into a probability

LR: Decision Boundary

- At the decision boundary where both classes are equiprobable

$$\begin{aligned} p(y=1|\boldsymbol{x},\boldsymbol{w}) &= p(y=0|\boldsymbol{x},\boldsymbol{w}) \\ \frac{\exp(\boldsymbol{w}^\top\boldsymbol{x})}{1+\exp(\boldsymbol{w}^\top\boldsymbol{x})} &= \frac{1}{1+\exp(\boldsymbol{w}^\top\boldsymbol{x})} \\ \exp(\boldsymbol{w}^\top\boldsymbol{x}) &= 1 \\ \boldsymbol{w}^\top\boldsymbol{x} &= 0 \end{aligned}$$

A linear hyperplane



- Very large positive $\boldsymbol{w}^\top\boldsymbol{x}$ means $p(y=1|\boldsymbol{w},\boldsymbol{x})$ close to 1
- Very large negative $\boldsymbol{w}^\top\boldsymbol{x}$ means $p(y=0|\boldsymbol{w},\boldsymbol{x})$ close to 1
- At decision boundary, $\boldsymbol{w}^\top\boldsymbol{x} = 0$ implies $p(y=1|\boldsymbol{w},\boldsymbol{x}) = p(y=0|\boldsymbol{w},\boldsymbol{x}) = 0.5$

MLE for Logistic Regression

Assumed 0/1, not -1/+1

- Likelihood (PMF of each input's label) is Bernoulli with prob $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$

$$p(y_n|\boldsymbol{w}, \boldsymbol{x}_n) = \text{Bernoulli}(\mu_n) = \mu_n^{y_n}(1-\mu_n)^{1-y_n}$$

- Overall likelihood, assuming i.i.d. observations

$$p(\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{X}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{w}, \boldsymbol{x}_n) = \prod_{n=1}^{N} \mu_n^{y_n}(1-\mu_n)^{1-y_n}$$

- The negative log-likelihood $NLL(\boldsymbol{w}) = -\log p(\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{X})$ simplifies to

"cross-entropy" loss (a popular loss function for classification)

Loss function

$$NLL(\boldsymbol{w}) = \sum_{n=1}^{N} -[y_n \log \mu_n + (1-y_n)\log(1-\mu_n)]$$

Very large loss if y_n close to 1and μ_n close to 0, or vice-versa

- Plugging in $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$ and simplifying

Good news: For LR, NLL is convex

$$\boxed{NLL(\boldsymbol{w}) = -\sum_{n=1}^{N} [y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n))]}$$

No closed-form expression for $\hat{\boldsymbol{w}}_{MLE} = \arg\min_{\boldsymbol{w}} NLL(\boldsymbol{w})$

Iterative opt needed (gradient or Hessian based). **Exercise:** Try working out the gradient of NLL and notice the expression's form

An Alternate Notation

- If we assume the label $\boldsymbol{y_n}$ as -1/+1 (not 0/1), the likelihood can be written as

$$p(y_n|\boldsymbol{w},\boldsymbol{x}_n) = \frac{1}{1+\exp(-y_n\boldsymbol{w}^\top\boldsymbol{x}_n)} = \sigma(y_n\boldsymbol{w}^\top\boldsymbol{x}_n)$$

- Slightly more convenient notation: A single expression gives the probabilities of both possible label values

- In this case, the total negative log-likelihood will be

$$NLL(\boldsymbol{w}) = \sum_{n=1}^{N} -\log p(y_n|\boldsymbol{w},\boldsymbol{x}_n) = \sum_{n=1}^{N} \log\left(1+\exp(-y_n\boldsymbol{w}^\top\boldsymbol{x}_n)\right)$$

MAP Estimation for Logistic Regression

- Need a prior on the weight vector $\boldsymbol{w} \in \mathbb{R}^D$
- Just like probabilistic linear regression, can use a zero-mean Gaussian prior

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \lambda^{-1}\boldsymbol{I}_D) \propto \exp\left(-\frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w}\right)$$

Or NLL – log of prior

- The MAP objective (log of posterior) will be log-likelihood + log of prior
- Therefore the MAP solution (ignoring terms that don't depend on $\boldsymbol{w}$) will be

$$\hat{\boldsymbol{w}}_{MAP} = \arg\min_{\boldsymbol{w}} NLL(\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w}$$

Good news: convex objective

- Just like MLE case, no closed form solution. Iterative opt methods needed
  - Highly efficient solvers (both first and second order) exist for MLE/MAP estimation for LR

Fully Bayesian Inference for Logistic Regression

- Doing fully Bayesian inference would require computing the posterior

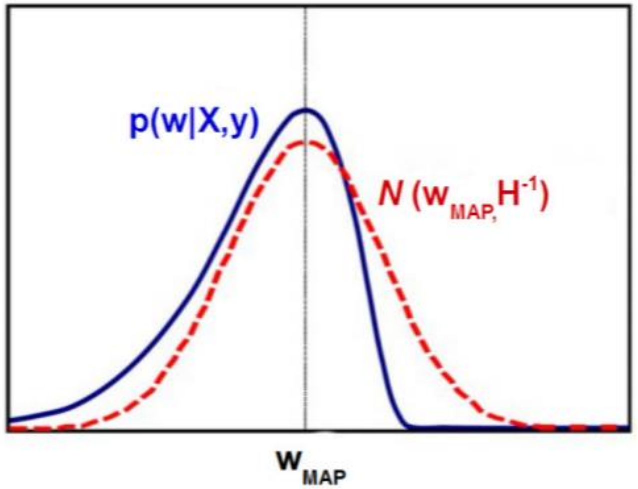
Gaussian

Bernoulli

$$p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\boldsymbol{w})p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})} = \frac{p(\boldsymbol{w})\prod_{n=1}^{N} p(y_n|\boldsymbol{w},\boldsymbol{x}_n)}{\int p(\boldsymbol{w})\prod_{n=1}^{N} p(y_n|\boldsymbol{w},\boldsymbol{x}_n)\, d\boldsymbol{w}}$$

Unfortunately, Gaussian and Bernoulli are not conjugate with each other, so analytic expression for the posterior can't be obtained unlike prob. linear regression

- Need to approximate the posterior in this case
- We will use a simple approximation called Laplace approximation

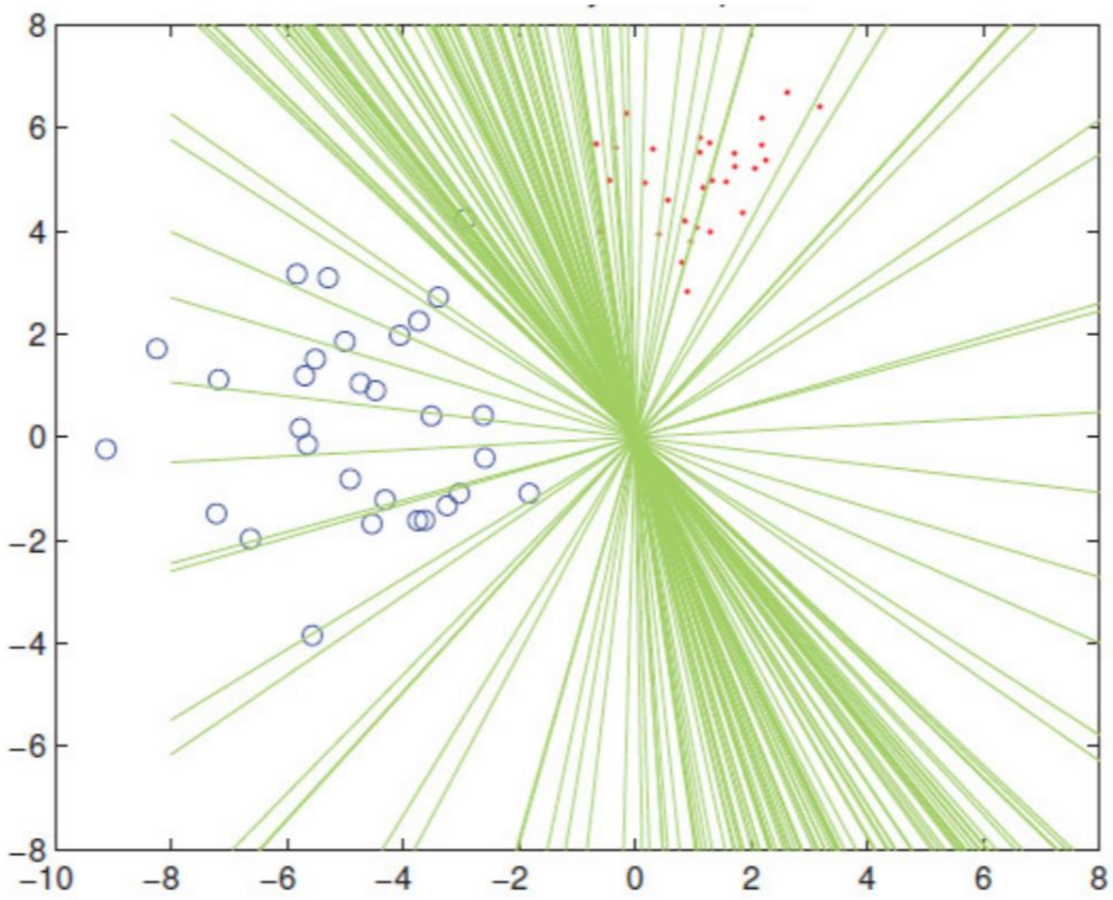


Approximates the posterior of $\boldsymbol{w}$ by a Gaussian whose mean is the MAP solution $\hat{\boldsymbol{w}}_{MAP}$ and covariance matrix is the inverse of the Hessian (Hessian: second derivative of the negative log-posterior of the LR model)

Can also employ more advanced posterior approximation methods, like MCMC and variational inference (beyond the scope of CS771)

Posterior for LR: An Illustration

- Can sample from the posterior of the LR model
- Each sample will give a weight vec defining a hyperplane separator



Not all separators are equally good; their goodness depends on their posterior probabilities

When making predictions, we can still use all of them but weighted by their importance based on their posterior probabilities

That's exactly what we do when computing the predictive distribution

Logistic Regression: Predictive Distribution

- When using MLE/MAP solution $\hat{\boldsymbol{w}}_{opt}$, can use the plug-in predictive distribution

$$p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(y_* = 1|\boldsymbol{w}, \boldsymbol{x}_*)p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})d\boldsymbol{w}$$

$$\approx p\left(y_* = 1\middle|\hat{\boldsymbol{w}}_{opt}, \boldsymbol{x}_*\right) = \sigma(\hat{\boldsymbol{w}}_{opt}{}^\top \boldsymbol{x}_n)$$

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \text{Bernoulli}[\sigma(\hat{\boldsymbol{w}}_{opt}{}^\top \boldsymbol{x}_n)]$$

- When using fully Bayesian inference, we must compute the posterior predictive

$$p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(y_* = 1|\boldsymbol{w}, \boldsymbol{x}_*)p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})d\boldsymbol{w}$$

Integral not tractable and must be approximated

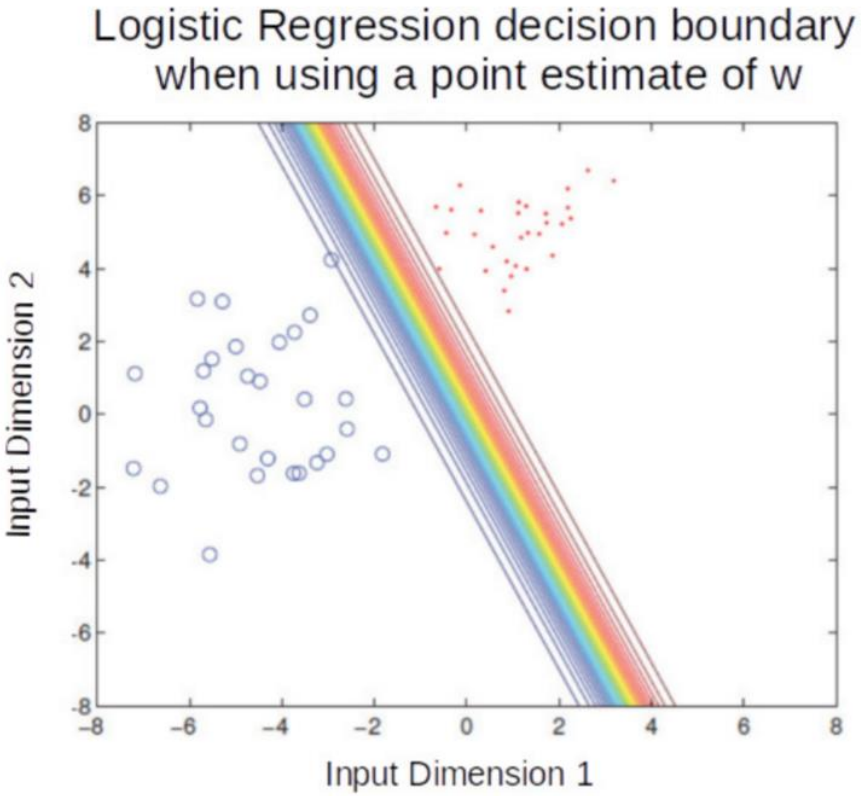
sigmoid

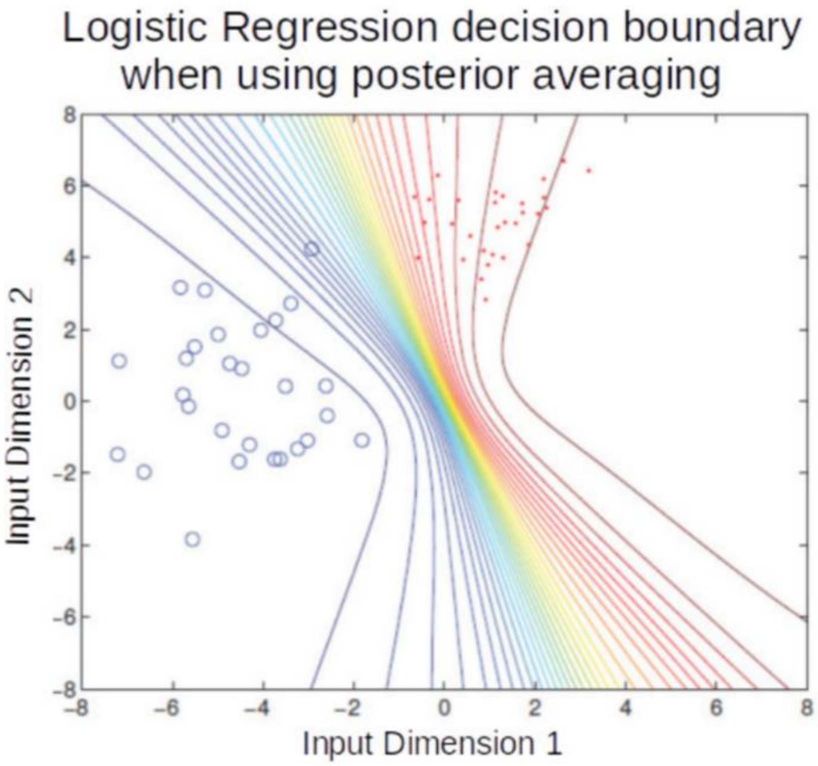
Gaussian (if using Laplace approx.)

Monte-Carlo approximation of this integral is one possible way

Generate M samples $\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_M$, from the Gaussian approx. of posterior and use $p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) \approx \frac{1}{M}\sum_{m=1}^{M} p(y_* = 1|\boldsymbol{w}_{\mathrm{m}}, \boldsymbol{x}_*) = \frac{1}{M}\sum_{m=1}^{M} \sigma(\boldsymbol{w}_m^\top \boldsymbol{x}_n)$

LR: Plug-in Prediction vs Postrerior Averaging





Posterior averaging is like using an ensemble of models. In this example, each model is a linear classifier but the ensemble-like effect resulted in nonlinear boundaries

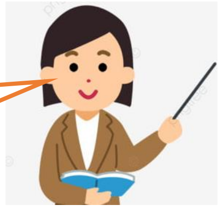
Multiclass Logistic (a.k.a. Softmax) Regression

- Also called multinoulli/multinomial regression: Basically, LR for $K > 2$ classes
- In this case, $y_n \in \{1,2,\dots,K\}$ and label probabilities are defined as

Softmax function

$$p(y_n = k|\boldsymbol{x}_n, \boldsymbol{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^{K} \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk}$$

Also note that $\sum_{\ell=1}^{K} \mu_{n\ell} = 1$ for any input $\boldsymbol{x}_n$



- K weight vecs $\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_K$ (one per class), each D-dim, and $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_K]$
- Each likelihood $p(y_n|\boldsymbol{x}_n, \boldsymbol{W})$ is a multinoulli distribution. Therefore total likelihood

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{W}) = \prod_{n=1}^{N} \prod_{\ell=1}^{K} \mu_{n\ell}^{y_{n\ell}}$$

Notation: $y_{n\ell} = 1$ if true class of $\boldsymbol{x}_n$ is ℓ and $y_{n\ell'} = 0 \; \forall \ell' \neq \ell$

- Can do MLE/MAP/fully Bayesian estimation for $\boldsymbol{W}$ similar to LR model

Coming up next

- Generative models for supervised learning

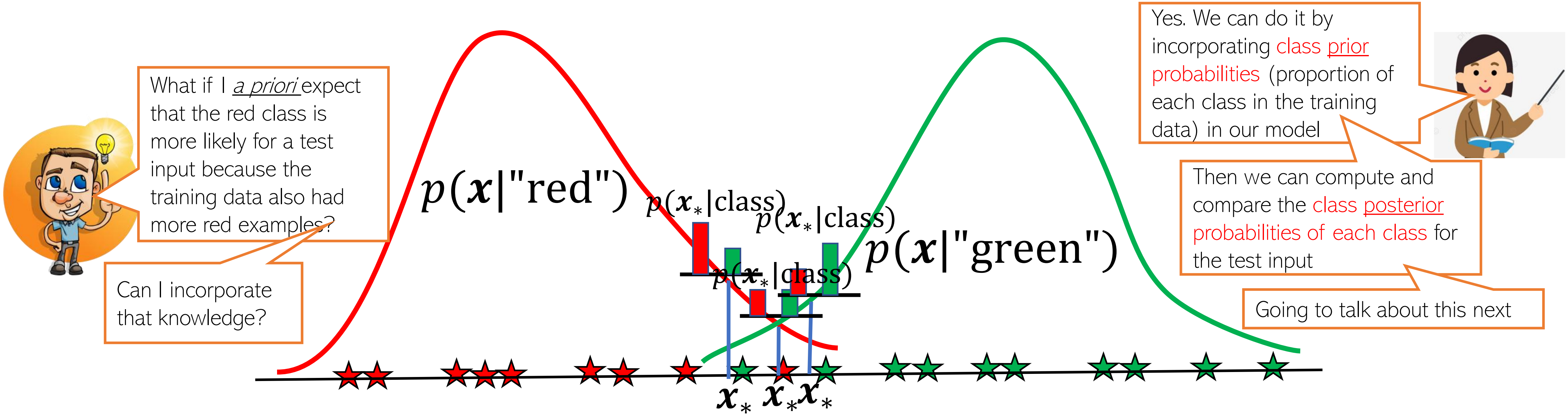
Probabilistic Models for Supervised Learning (3): Generative Classification and Regression

CS771: Introduction to Machine Learning

Piyush Rai

Generative Classification: A Basic Idea

- Learn the probability distribution of inputs from each class ("class-conditional")


$p(\boldsymbol{x}|\text{"red"})$
$p(\boldsymbol{x}_*|\text{class})$
$p(\boldsymbol{x}_*|\text{class})$
$p(\boldsymbol{x}_*|\text{class})$
$p(\boldsymbol{x}|\text{"green"})$
$\boldsymbol{x}_*$ $\boldsymbol{x}_*$ $\boldsymbol{x}_*$

What if I *a priori* expect that the red class is more likely for a test input because the training data also had more red examples?

Can I incorporate that knowledge?

Yes. We can do it by incorporating class prior probabilities (proportion of each class in the training data) in our model

Then we can compute and compare the class posterior probabilities of each class for the test input

Going to talk about this next

- Usually assume some form (e.g., Gaussian) and estimate the parameters of that distribution (using MLE/MAP/fully Bayesian approach)
- Predict label of a test input $\boldsymbol{x}_*$ by comparing its probabilities under each class
  - Or can report the probability of belonging to each class (soft prediction)

Generative Classification: More Generally..

- Consider a classification problem with $K \geq 2$ classes
- The class prior probability of each class $k \in \{1,2,\ldots,K\}$ is $p(y=k)$
- Can use Bayes rule to compute class posterior probability for a test input $\boldsymbol{x}_*$

Roughly speaking, what's the fraction of each class in the training data

$$p(y_* = k|\boldsymbol{x}_*, \theta) = \frac{p(\boldsymbol{x}_*, y_* = k|\theta)}{p(\boldsymbol{x}_*|\theta)} = \frac{p(y_* = k|\theta)p(\boldsymbol{x}_*|y_* = k, \theta)}{p(\boldsymbol{x}_*|\theta)}$$

Class prior distribution for class k

Class-conditional distribution of inputs from class k

This is just the marginal distribution of the joint distribution in the numerator (summed over all K values of y_*)

θ collectively denotes the parameters the joint distribution of inputs and labels depends on

Setting $p(y_* = k|\theta) = 1/K$ will give us the approach that predicts by comparing the probabilities $p(\boldsymbol{x}_*|y_* = k, \theta)$ of $\boldsymbol{x}_*$ under each of the classes

- We will first estimate the parameters of class prior and class-conditional distributions. Once estimated, we can use the above rule to predict the label for any test input
  - Can use MLE/MAP/fully Bayesian approach. We will only consider MLE/MAP here

Estimating Class Priors

Note: Can also do MAP estimation using a Dirichlet prior on π (this is akin to using Beta prior for doing MAP estimation for the bias of a coin). May try this as an exercise

- Estimating class priors $p(y = k)$ is usually straightforward in gen. classification
- Roughly speaking, it is the proportion of training examples from each class
  - Note: The above is true only when doing MLE (as we will see shortly)
  - If estimating class priors using MAP/fully Bayesian, they will be a "smooth version" of the proportions (because of the effect of regularization)
- The class prior distribution is assumed to be a discrete distribution (multinoulli)

$\pi_k = p(y = k)$

These probabilities sum to 1: $\sum_{k=1}^{K} \pi_k = 1$

Generalization of Bernoulli

$$p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\pi_1, \pi_2, \dots, \pi_K) = \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y=k]}$$

- Given N i.i.d. labelled examples $\{(x_n, y_n)\}_{n=1}^{N}$, $y_n \in \{1,2,\dots,K\}$ the MLE soln

$$\boldsymbol{\pi}_{MLE} = \arg\max_{\boldsymbol{\pi}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{\pi})$$

Subject to constraint $\sum_{k=1}^{K} \pi_k = 1$

Can use Lagrange based opt. (note that we have an equality constraint)

Exercise: Verify that the MLE solution will be $p(y = k) = \pi_k = N_k/N$ where $N_k = \sum_{n=1}^{N} \mathbb{I}[y = k]$ (the frac. of class k examples)

Estimating Class-Conditionals

To be estimated using inputs from class k

- Can assume an appropriate distribution $p(\boldsymbol{x}|y = k, \theta)$ for inputs of each class
- If $\boldsymbol{x}$ is D-dim, it will be a D-dim. distribution. Choice depends on various factors
  - Nature of input features, e.g.,
    - If $\boldsymbol{x} \in \mathbb{R}^D$, can use a D-dim Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
    - If $\boldsymbol{x} \in \{0,1\}^D$, can use D Bernoullis (one for each feature)
    - Can also choose more flexible/complex distributions if possible to estimate
  - Amount of training data available
    - With little data from a class, difficult to estimate the params of its class-cond. distribution

Some workarounds: Use strong regularization, or a simple form of the class-conditional (e.g., use a spherical/diagonal rather than a full covariance if the class-cond is Gaussian), or assume features are independent given class ("naïve Bayes" assumption)

A big issue especially if the number of features (D) is very large

- Once decided the form of class-cond, estimate $\boldsymbol{\theta}$ via MLE/MAP/Bayesian infer.
  - This essentially is a density estimation problem for the class-cond.
  - In principle, can use any density estimation method

Gen. Classifn. using Gaussian Class-conditionals

- The generative classification model $p(y=k|\boldsymbol{x}) = \dfrac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x}|\theta)}$
- Assume each class-conditional $p(\boldsymbol{x}|y=k)$ to be a Gaussian

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}_k|}}\exp[-(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)]$$

A benefit of modeling each class by a distribution (recall that LwP had issues)

Since the Gaussian's covariance models its shape, we can learn the shape of each class ☺

- Class prior is multinoulli (we already saw): $p(y=k)=\pi_k, \pi_k \in (0,1), \sum_{k=1}^{K}\pi_k = 1$
- Let's denote the parameters of the model collectively by $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$
  - Can estimate these using MLE/MAP/Bayesian inference
  - Already saw the MLE solution for $\boldsymbol{\pi}$: $\pi_k = N_k/N$ (can also do MAP)
  - MLE solution for $\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{y_n=k}\boldsymbol{x}_n$, $\boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{y_n=k}(\boldsymbol{x}_n-\boldsymbol{\mu}_k)(\boldsymbol{x}_n-\boldsymbol{\mu}_k)^\top$

Can also do MAP estimation for $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ using a Gaussian prior on $\boldsymbol{\mu}_k$ and inverse Wishart prior on $\boldsymbol{\Sigma}_k$

Exercise: Try to derive this. I will provide a separate note containing the derivation

- If using point est (MLE/MAP) for $\boldsymbol{\theta}$, predictive distribution will be

$$p(y_*=k|\mathbf{x}_*,\theta) = \frac{\pi_k|\boldsymbol{\Sigma}_k|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}_*-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_*-\boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K}\pi_k|\boldsymbol{\Sigma}_k|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}_*-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_*-\boldsymbol{\mu}_k)\right]}$$

Can predict the most likely class for the test input $\boldsymbol{x}_*$ by comparing these probabilities for all values of k

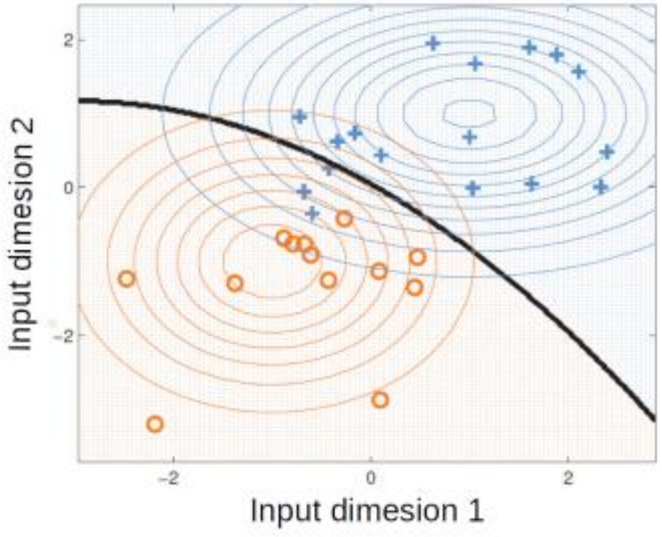
Note that the exponent has a Mahalanobis distance like term. Also, accounts for the fraction of training examples in class k

Decision Boundary with Gaussian Class-Conditional

- As we saw, the prediction rule when using Gaussian class-conditional

$$p(y=k|\mathbf{x},\theta)=\frac{\pi_k|\boldsymbol{\Sigma}_k|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}{\sum_{k=1}^K\pi_k|\boldsymbol{\Sigma}_k|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}$$

- The decision boundary between any pair of classes will be a quadratic curve



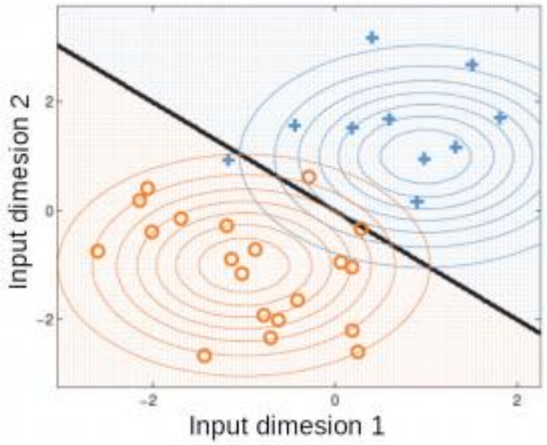
Reason: For any two classes k and k' at the decision boundary, we will have $p(y=k|x,\theta)=p(y=k'|x,\theta)$. Comparing their logs and ignoring terms that don't contain $\boldsymbol{x}$, can easily see that

$$(\mathbf{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)-(\mathbf{x}-\boldsymbol{\mu}_{k'})^\top\boldsymbol{\Sigma}_{k'}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{k'})=0$$

Decision boundary contains all inputs $\boldsymbol{x}$ that satisfy the above This is a quadratic function of $\boldsymbol{x}$ (this model is sometimes referred to Quadratic Discriminant Analysis)

Decision Boundary with Gaussian Class-Conditional

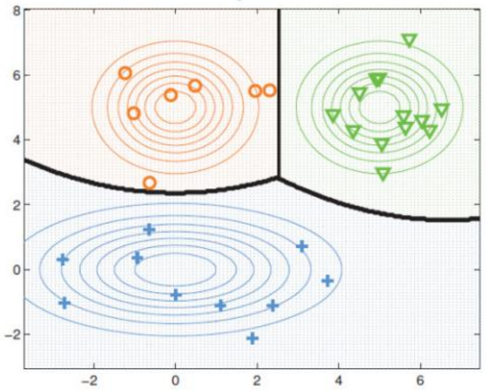
- Assume all classes are modeled using the same covariance matrix $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$
- In this case, the decision boundary b/w any pair of classes will be linear

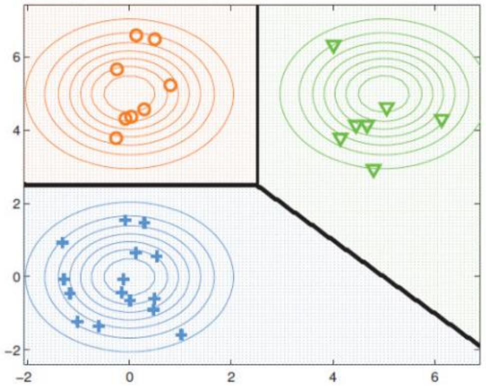


Reason: Again using $p(y = k|x, \theta) = p(y = k'|x, \theta)$, comparing their logs and ignoring terms that don't contain $\boldsymbol{x}$, we have

$$(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) - (\mathbf{x} - \boldsymbol{\mu}_{k'})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{k'}) = 0$$

Quadratic terms of $\boldsymbol{x}$ will cancel out; only linear terms will remain; hence decision boundary will be a linear function of $\boldsymbol{x}$ (Exercise: Verify that we can indeed write the decision boundary between this pair of classes as $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ where $\boldsymbol{w}$ and b depend on $\boldsymbol{\mu}_k, \boldsymbol{\mu}_{k'}$ and $\boldsymbol{\Sigma}$)





If we assume the covariance matrices of the assumed Gaussian class-conditionals for any pair of classes to be equal, then the learned separation boundary b/w this pair of classes will be linear; otherwise, quadratic as shown in the figure on left

A Closer Look at the Linear Case

- For the linear case (when $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$), the class posterior probability

$$p(y = k|\mathbf{x}, \theta) \propto \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

- Expanding further, we can write the above as

$$p(y = k|\boldsymbol{x}, \theta) \propto \exp\left[\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k\right] \exp\left[\boldsymbol{x}^\top \Sigma^{-1}\boldsymbol{x}\right]$$

- Therefore, the above class posterior probability can be written as

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\exp\left[\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right]}{\sum_{k=1}^K \exp\left[\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right]}$$

$$\boldsymbol{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k \qquad b_k = -\tfrac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k$$

If all Gaussians class-cond have the same covariance matrix (basically, of all classes are assumed to have the same shape)

- The above has *exactly* the same form as softmax classification (thus softmax is a special case of a generative classification model with Gaussian class-conditionals)

A Very Special Case: LwP Revisited

- Note the prediction rule when $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$

$$\begin{aligned}\hat{y} = \arg\max_k p(y = k|\mathbf{x}) &= \arg\max_k \quad \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right] \\ &= \arg\max_k \quad \log \pi_k - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\end{aligned}$$

- Also assume all classes to have equal no. of training examples, i.e., $\pi_k = 1/K$. Then

$$\hat{y} = \arg\min_k \quad (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

The Mahalanobis distance matrix = $\boldsymbol{\Sigma}^{-1}$

- Equivalent to assigning $\boldsymbol{x}$ to the “closest” class in terms of a Mahalanobis distance
- If we further assume $\boldsymbol{\Sigma} = \boldsymbol{I}_D$ then the above is exactly the LwP rule

Generative Classification: Some Comments

- A simple but powerful approach to probabilistic classification
- Especially easy to learn if class-conditionals are simple
  - E.g., Gaussian with diagonal covariances ⇒ Gaussian naïve Bayes
  - Another popular model is multinomial naïve Bayes (widely used for document classification)
  - The naïve Bayes assumption: features are conditional independent given class label

$$p(\boldsymbol{x}|y=k)=\prod_{d=1}^{D} p(x_d|y=k)$$

Benefit: Instead of estimating a D-dim distribution which may be hard (if we don't have enough data), we will estimate D one-dim distributions (much simpler task)

- Can choose the form of class-conditionals $p(\boldsymbol{x}|y=k)$ based on the type of inputs $\boldsymbol{x}$

Will see such methods later

- Can handle missing data (e.g., if some part of the input $\boldsymbol{x}$ is missing) or missing labels

Will see such methods later

- Generative models are also useful for unsup. and semi-sup. learning

Generative Models for Regression

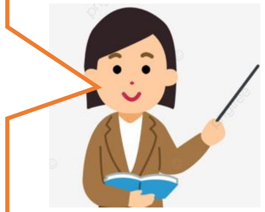
- Yes, we can even model regression problems using a generative approach
- Note that the output y is not longer discrete (so no notion of a class-conditional)
- However, the basic rule of recovering a conditional from joint would still apply

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

- Thus we can model the joint distribution $p(\boldsymbol{x},y|\theta)$ of features $\boldsymbol{x}$ and outputs $y \in \mathbb{R}$
  - If features are real-valued the we can model $p(\boldsymbol{x},y|\theta)$ using a $(D+1)$-dim Gaussian
  - From this $(D+1)$-dim Gaussian, we can get $p(y|\boldsymbol{x},\theta)$using Gaussian conditioning formula
  - If joint is Gaussian, any subset of variables (y here), given the rest ($\boldsymbol{x}$ here) is also a Gaussian!
  - Refer to the Gaussian results from maths refresher slides for the result

Discriminative vs Generative

Proponents of discriminative models: Why bother modeling $\boldsymbol{x}$ if $\boldsymbol{y}$ is what you care about? Just model $\boldsymbol{y}$ directly instead of working hard to model x by learning the class-conditional



- Recall that discriminative approaches model $\boldsymbol{p(y|x)}$ directly
- Generative approaches model $\boldsymbol{p(y|x)}$ via $\boldsymbol{p(x,y)}$
- Number of parameters: Discriminative models have fewer parameters to be learned
  - Just the weight vector/matrix $\boldsymbol{w}/\boldsymbol{W}$ in case of logistic/softmax classification
- Ease of parameter estimation: Debatable as to which one is easier
  - For “simple” class-conditionals, easier for gen. classifn model (often closed-form solution)
  - Parameter estimation for discriminative models (logistic/softmax) usually requires iterative methods(although objective functions usually have global optima)
- Dealing with missing features: Generative models can handle this easily
  - E.g., by integrating out the missing features while estimating the parameters)
- Inputs with features having mixed types: Generative model can handle this
  - Appropriate $\boldsymbol{p(x_d|y)}$ for each type of feature in the input. Difficult for discriminative models

Discriminative vs Generative (Contd)

- Leveraging unlabeled data: Generative models can handle this easily by treating the missing labels are latent variables and are ideal for Semi-supervised Learning. Discriminative models can't do it easily

- Adding data from new classes: Discriminative model will need to be re-trained on all classes all over again. Generative model will just require estimating the class-cond of newly added classes

- Have lots of labeled training data? Discriminative models usually work very well
- Final Verdict? Despite generative classification having some clear advantages, both methods can be quite powerful (the actual choice may be dictated by the problem)
  - Important to be aware of their strengths/weaknesses, and also the connections between these
- Possibility of a Hybrid Design? Yes, Generative and Disc. models can be combined, e.g.,
  - "Principled Hybrids of Generative and Discriminative Models" (Lassere et al, 2006)
  - "Deep Hybrid Models: Bridging Discriminative & Generative Approaches" (Kuleshov & Ermon, 2017)

Coming up next

- Large-margin hyperplane based classifiers (support vector machines)
- Kernel methods for learning nonlinear models