Probabilistic ML

Based on Dr. Piyush Rai's slides from ML course at IITK

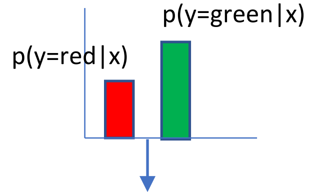
And

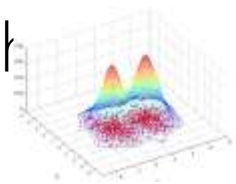
"Introduction to Statistical Learning" Book 2nd edition, Chapter 4

https://www.statlearning.com/

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 $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N)\}$ estimate th  itional $p(y|\boldsymbol{x})$
  - Given inputs $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$, estimate the distribution $p(\boldsymbol{x})$ of the inputs
  - Note 1: These dist. will depend on some parameters θ (to be estimated), and written as

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

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

Probabilistic Modeling: The Basic Idea

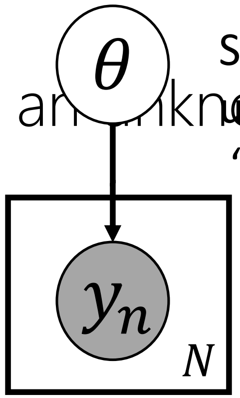
- Assume N observations $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$, generated from a presumed prob. model

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

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

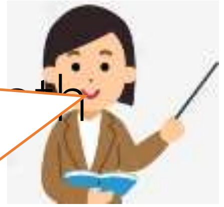
- Here $p(y|\theta)$ is a conditional distribution, conditioned on params θ (to be learne
  - N own or an unknown random v oth
  cas

The parameters θ 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 y_* is. Example: if y_* denotes the outcome of a coin toss, then what is $p(y_* = \text{"}head\text{"}|\boldsymbol{y})$, given N previous coin tosses $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$



- Some of the tasks that we may be interested in
  - Parameter estimation: Estimating the unknown parameters θ (and other unknowns

Parameter Estimation in Probabilistic Models

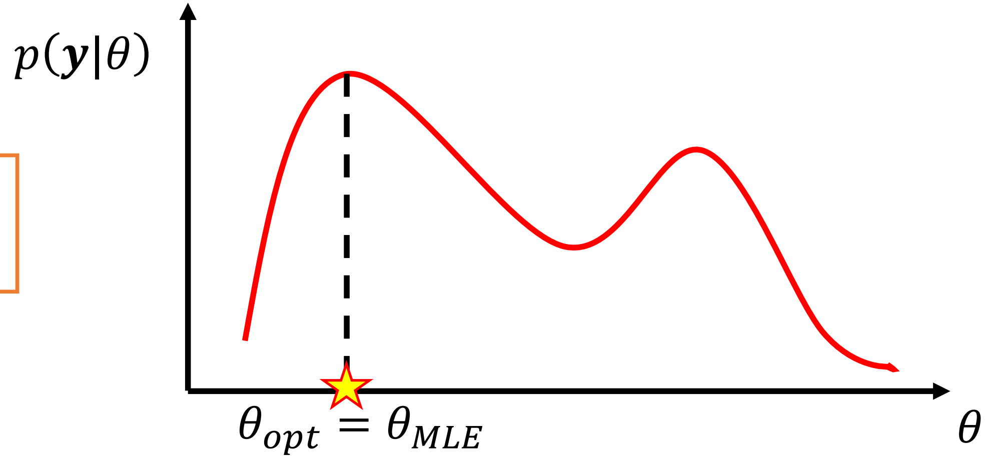
- Since data is assumed to be i.i.d., we can write down its total proba

This now is an optimization problem essentially (θ being the unknown)

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

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

How do I find the best θ ?

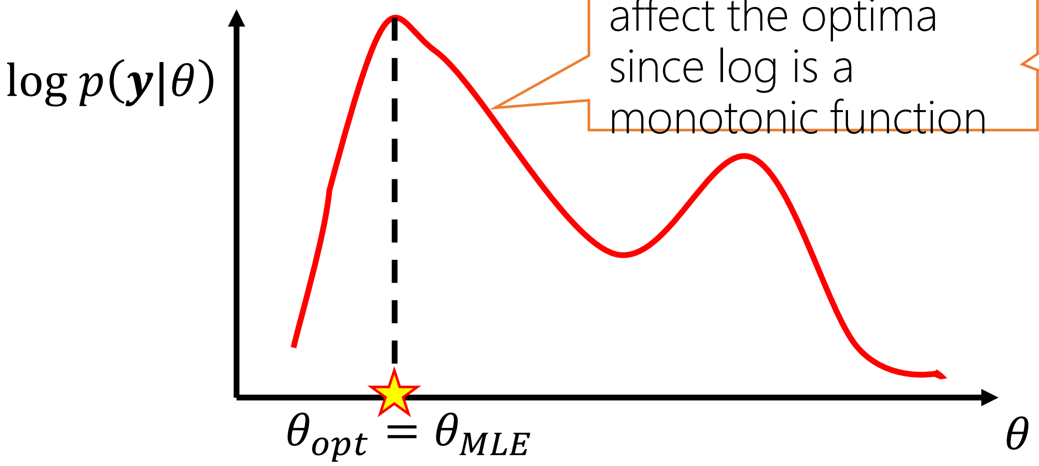


Well, one option is to find the θ that maximizes the likelihood (probability of the observed data) – basically, which value of θ makes the observed data most likely to have come from the assumed distribution $p(y|\theta)$ --- Maximum Likelihood Estimation (MLE)

- 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

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)

Negative Log-Likelihood (NLL)

- The MLE problem can also be easily written as a minimization problem

$$\theta_{MLE} = \arg\max_{\theta} \sum_{n=1}^{N} \log p(y_n|\theta) = \arg\min_{\theta} \sum_{n=1}^{N} -\log p(y_n|\theta)$$

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

$$\boxed{\theta_{MLE} = \arg\min_{\theta} NLL(\theta)}$$

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)

- NLL is analogous to a loss function
  - The negative log-lik $(-\log p(y_n|\theta))$ is akin to the loss on each data

Such priors have various other benefits as we will see later

- Thus doing MLE is akin to minimizing training

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

MLE: An Example

- Consider a sequence of N coin toss outcomes (observations)

Probability of a head

- 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 $\theta \in (0,1)$

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

- Here θ the unknown param (probability of head). Want to estimate it using MLE

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

- Log-likelihood: $\sum_{n=1}^{N} \log p(y_n|\theta) = \sum_{n=1}^{N} [y_n \log\theta + (1-y_n)\log(1-\theta)]$
- ...imizing NLL) w.r.t. ...e

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

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

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!

Refresher

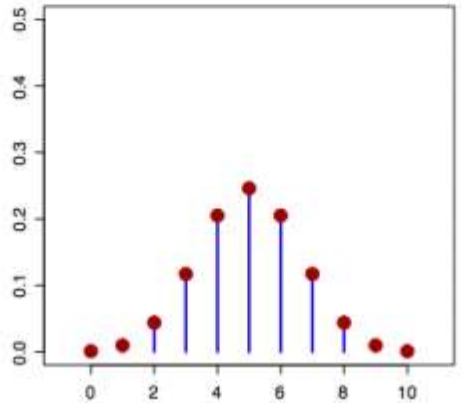
Random Variables

- Informally, a random variable (r.v.) X denotes possible outcomes of an event

- Can be discrete (i.e., finite many possible outcomes)

- 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

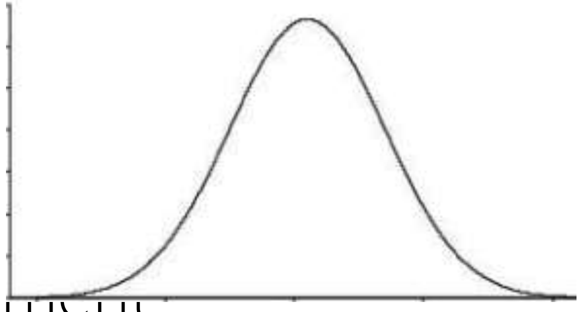
$p(X)$



X (a discrete r.v.)

- 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

$p(X)$

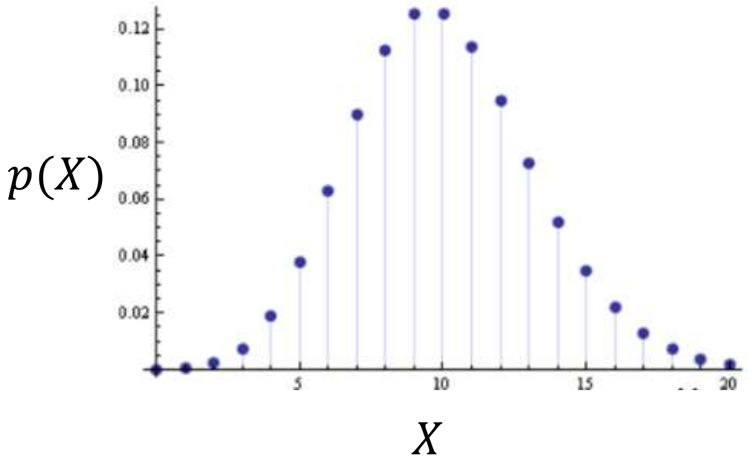


X (a continuous r.v.)

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 <u>value</u> of the PMF at x

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



Continuous Random Variables

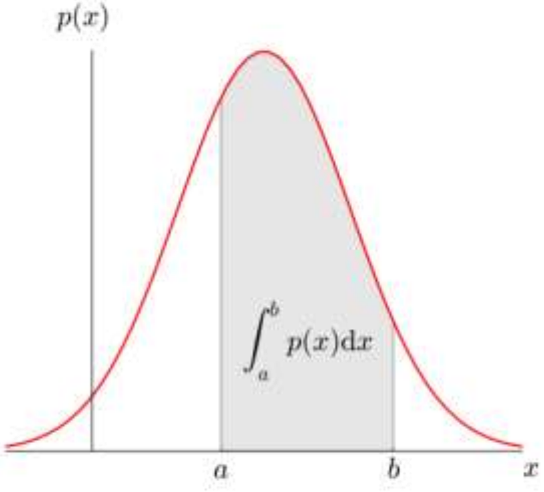
- For a continuous r.v. X, a *probability* $p(X = x)$ or $p(x)$ is meaningless
- For cont. r.v., we talk in terms of prob. within an <u>interval</u> $X \in (x, x + \delta x)$
  - $p(x)\delta x$ is the prob. that $X \in (x, x + \delta x)$ as $\delta x \to 0$
  - $p(x)$ is the probability density at $X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

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

$$p(x) \geq 0$$

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

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



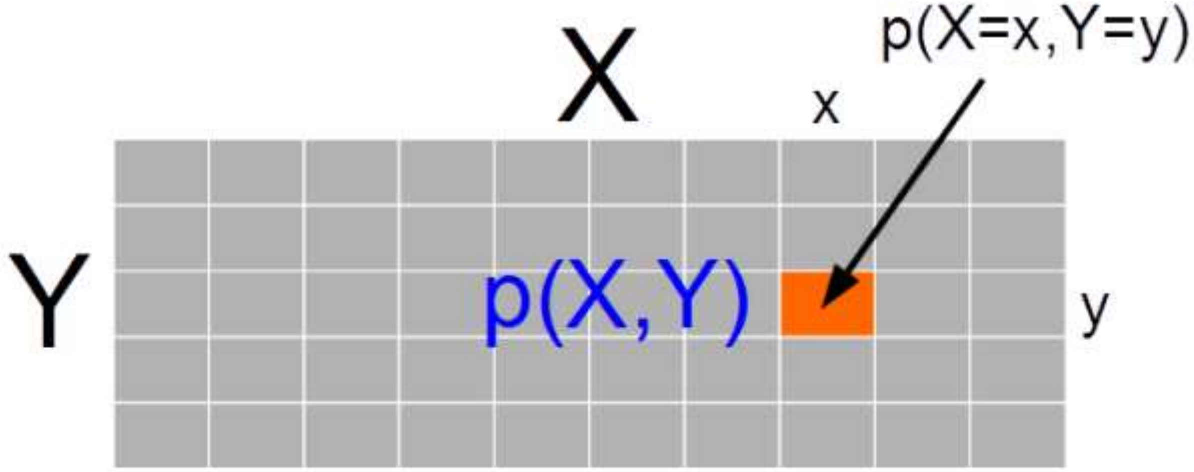
A word about notation

- $p(.)$ can mean different things depending on the context
- $p(X)$ denotes the distribution (PMF/PDF) of an r.v. 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



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

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

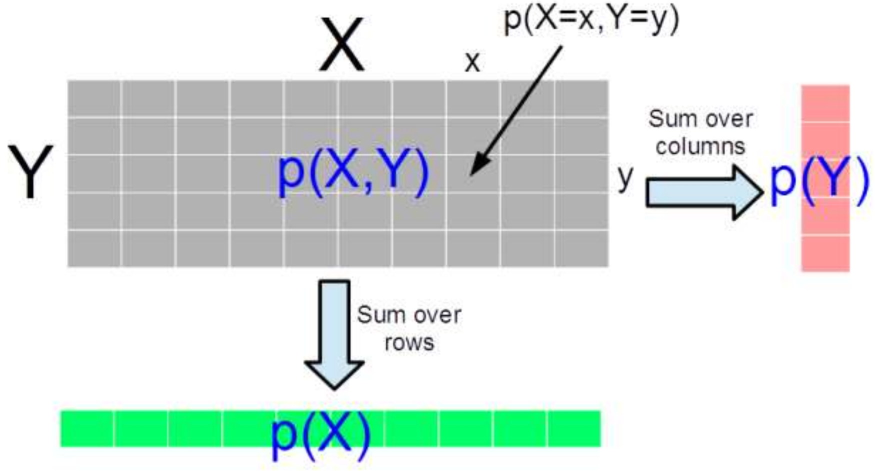
- For two conti ... have jo

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

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 discr PM



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"

$$p(X) = \int_y p(X, Y = y)dy, \quad p(Y) = \int_x p(X = x, Y)dx$$

- For continuous r.v.'s,

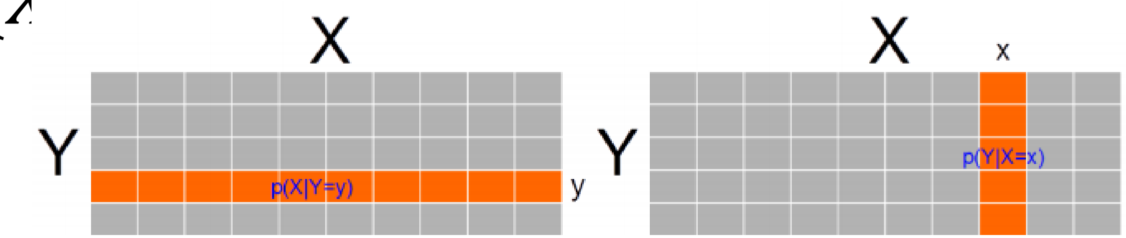
Conditional Probability Distribution

- Consider two r.v.'s X and Y (discrete/continuous – both need not of same type)

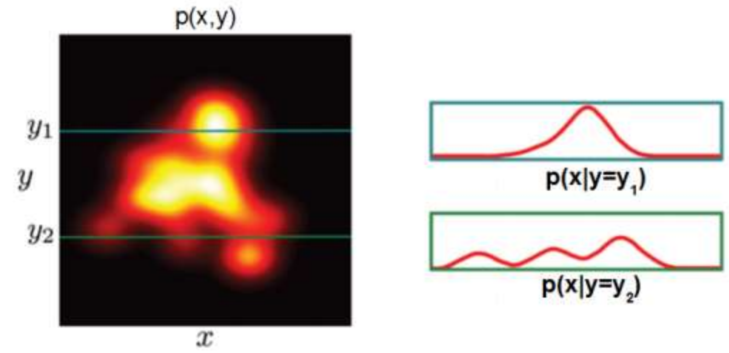
- Conditional PMF/PDF $p(X|Y)$ is the prob. dist. of one r.v. X, fixing other r.v. Y

- $p(X|Y=y)$ or $p(Y|X=x)$ like taking a s ... X, Y)

Discrete Random Variables



Continuous Random Variables



- Note: A conditional PMF/PDF may also be conditi... that is not the value of an r.v. but some fixed quan... general

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

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 proc

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

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

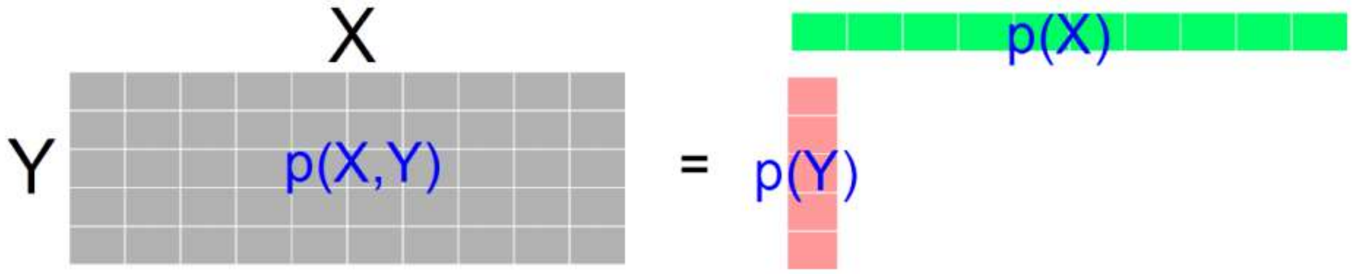
  For continuous r.v.: $p(Y|X) = \frac{p(X|Y)p(Y)}{\int_Y p(X|Y)p(Y)dY}$

- **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

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

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 $p(X)$

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

Probability that $X = x$

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

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

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 $Z = X + Y$

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

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

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

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

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

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

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

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 $\mathbb{E}[X] = \mu$

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

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

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

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

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

Important result

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

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

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

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

- 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

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

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

Common Probability Distributions

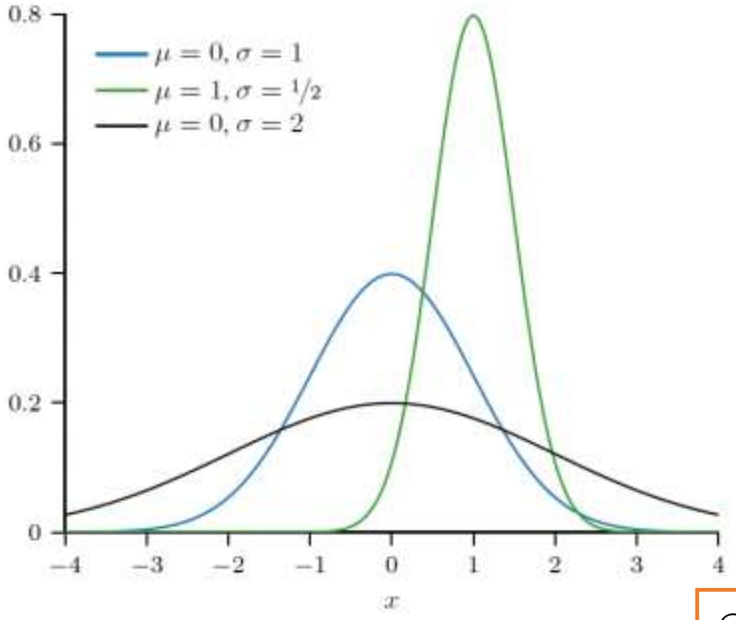
Important: We will use these extensively to model data as well as parameters of models

- 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

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

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



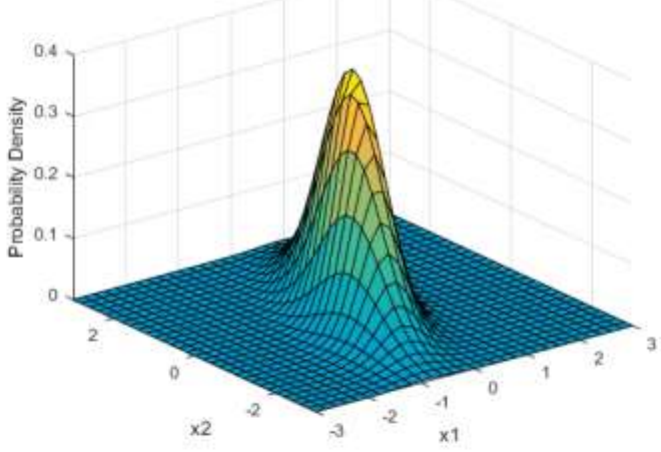
- Mean: $\mathbb{E}[x] = \mu$
- Variance: $\text{var}[x] = \sigma^2$
- Inverse of variance is called precision: $\beta = \frac{1}{\sigma^2}$. $\mathcal{N}(x|\mu,\beta) = \sqrt{\frac{\beta}{2\pi}} \exp\left[-\frac{\beta}{2}(x-\mu)^2\right]$

Gaussian PDF in terms of precision

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 $\boldsymbol{\Sigma}$

A two-dimensional Gaussian

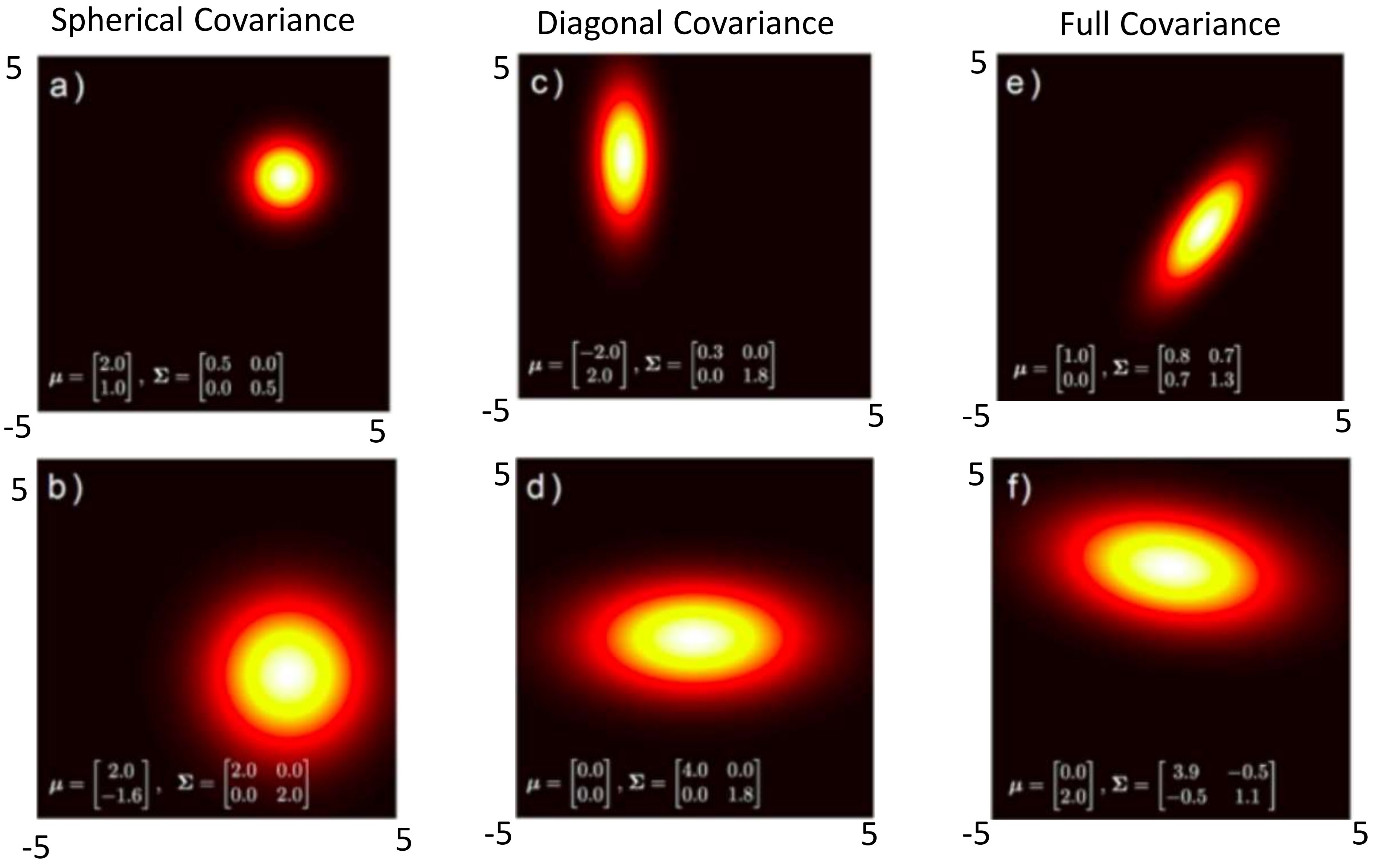


$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}|}}\exp[-(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})]$$

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

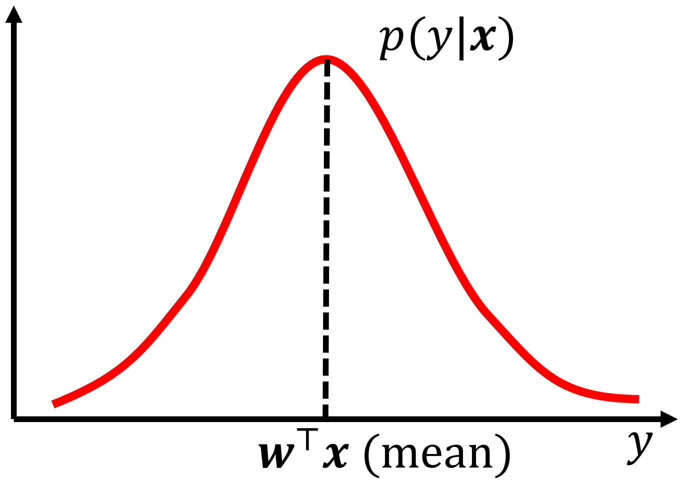
Diagonal: Unequal spreads (variances) along all directions but still axis-parallel

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

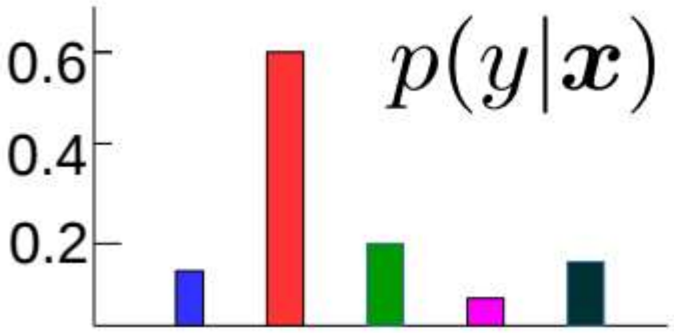
Probabilistic Models for Supervised Learning

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

Probabilistic Linear Regression



Probabilistic Classification



- $p(y|\boldsymbol{x})$ is more informative than a single prediction $\boldsymbol{y}$
  - From $p(y|\boldsymbol{x})$, 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 $p(y|\boldsymbol{x})$)

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

Probabilistic Models for Supervised Learning

- Usually two ways to model the conditional distribution $p(y|\boldsymbol{x})$
- Approach 1: Don't model $\boldsymbol{x}$, and model $p(y|\boldsymbol{x})$ <u>directly</u> 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 y and use these to get the conditional as

"generative" sup learning

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

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

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

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

Probabilistic Linear Regression

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

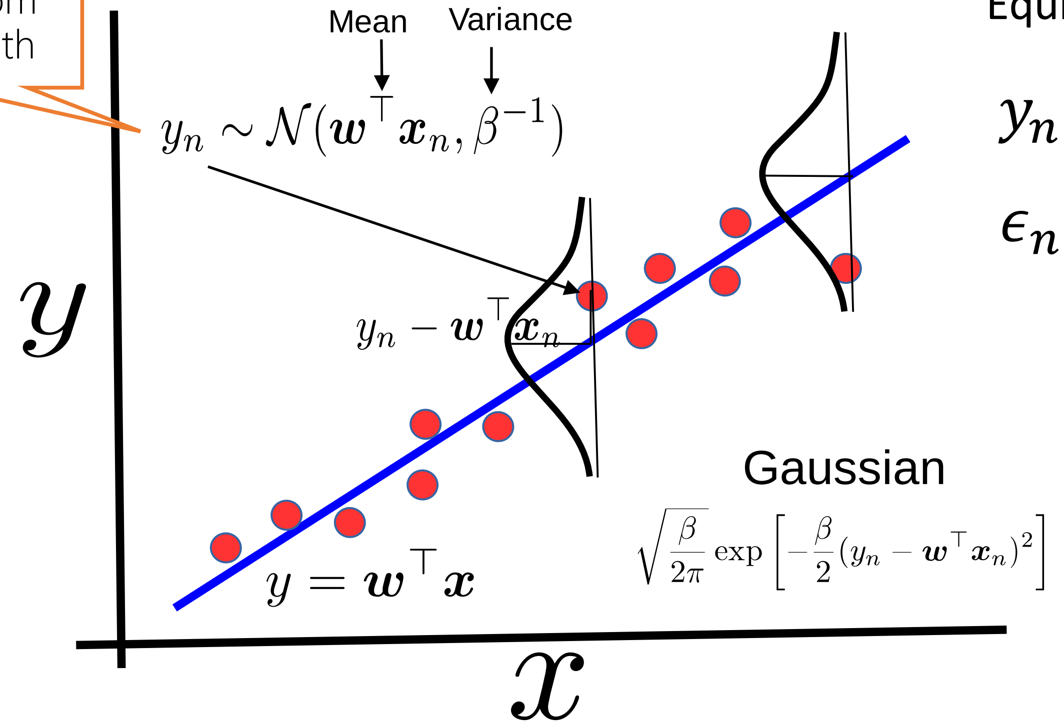
Gaussian distribution

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

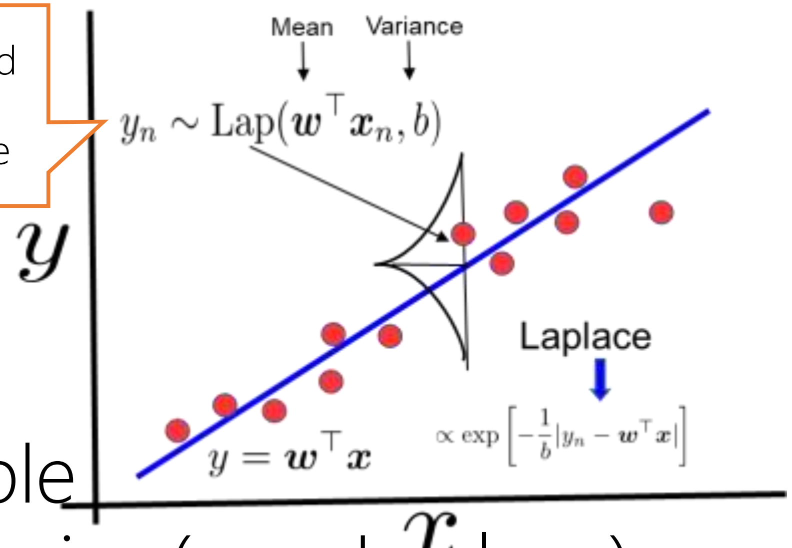
Equivalently:

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

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

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

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

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

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

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

- 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

Classification Motivation

- A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions.
  - {Stroke, Drug Overdose, Epileptic Seizure}
  - Which of the three conditions does the individual have?
- An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user’s IP address, past transaction history, and so forth.
- On the basis of DNA sequence data for a number of patients with and without a given disease, a biologist would like to figure out which DNA mutations are deleterious (disease-causing) and which are not.

Why not linear regression for Classification

- No ordering possible
  - Say, we encode: Stroke=1, Drug Overdose=2, Epileptic Seizure =3
  - Is “drug overdose” midway between “Stroke” and Seizure?
- Linear Regression predictions can be –ve and also outside the desired range for some values of input. How can we control that?

Logistic Regression

- Let's take a dataset: Loan default
- We wish to compute $P[Default = Yes|Balance = X]$

We may model,
$$p(X) = \beta_0 + \beta_1 X$$

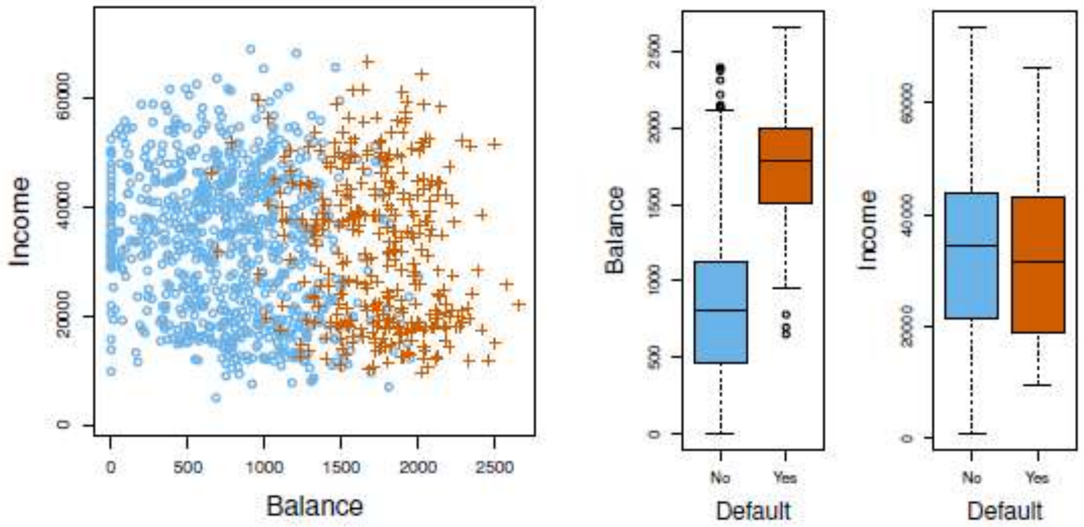
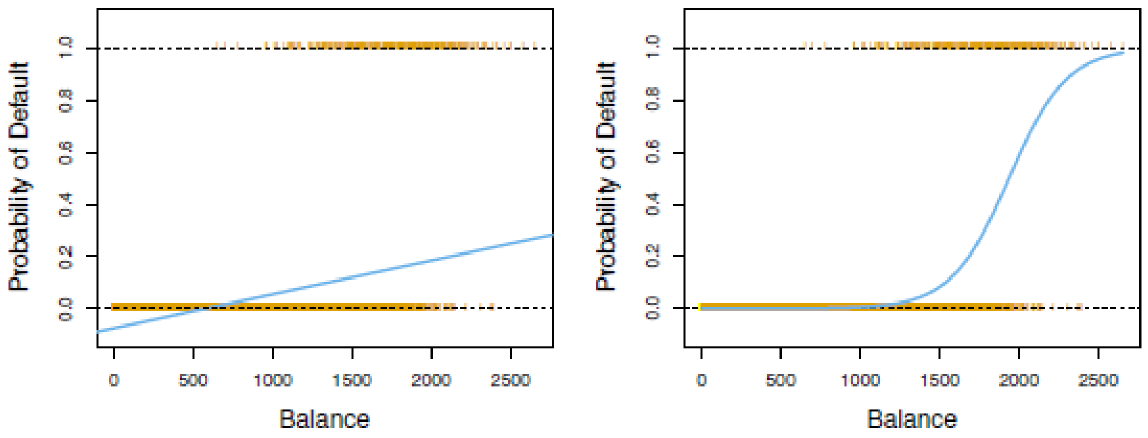


FIGURE 4.1. *The* Default *data set. Left: The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue. Center: Boxplots of* balance *as a function of* default *status.* Right: *Boxplots of* income *as a function of* default *status.*

What is the problem with this approach?

Logistic Function/Odds/LogOdds/Logit



- $p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+ e^{\beta_0+\beta_1 X}}$ [Logistic Function]
- Odds = $\frac{p(x)}{1- p(x)} = e^{\beta_0+\beta_1 X}$
- $\log\left(\frac{p(x)}{1- p(x)}\right) = \beta_0 + \beta_1 X$ [Log-odds/Logit]

Logistic Regression (LR)

Multi-class extension known as "softmax regression"

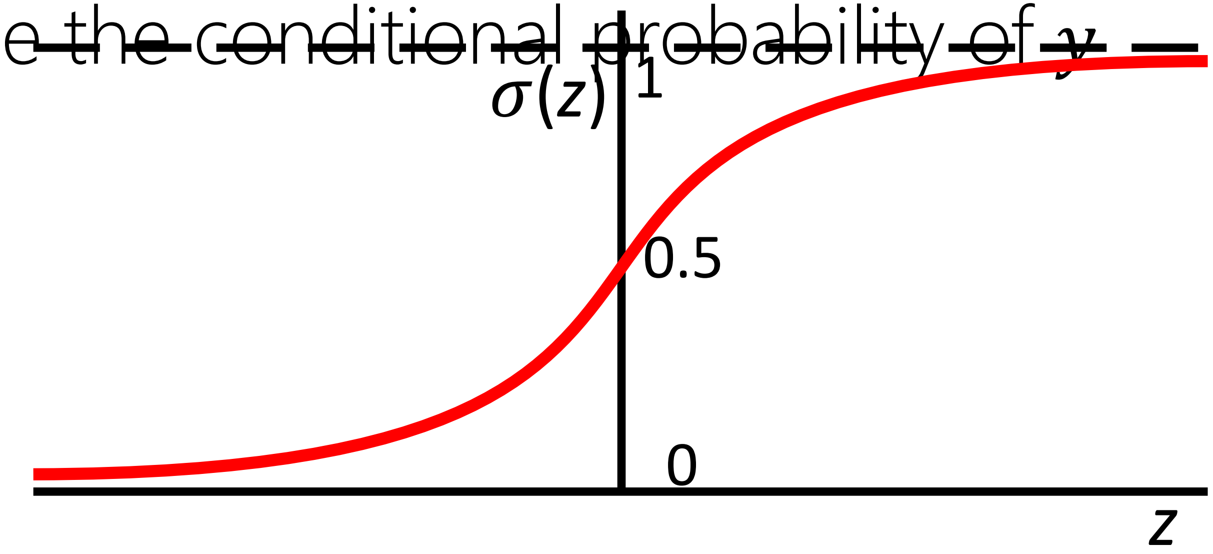
Both very widely used

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



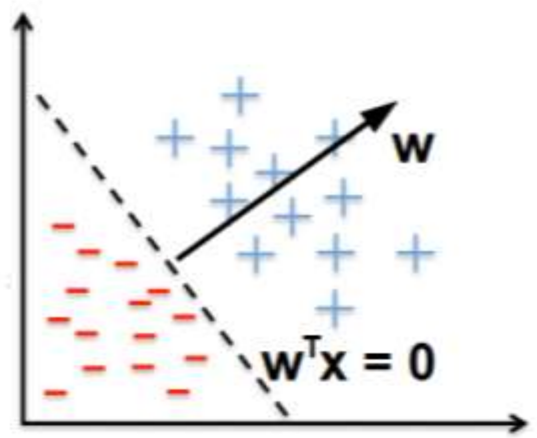
- Here $\boldsymbol{w}^\top \boldsymbol{x}$ is the score for input $\boldsymbol{x}$. The sigmoid turns it into a

LR: Decision Boundary

- At the decision boundary where both classes are equiprobable

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

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

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

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

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

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

Good news: For LR, NLL is convex

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

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

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

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

Predicting Defaulters!!

	Coefficient	Std. error	z-statistic	p-value
Intercept	−10.6513	0.3612	−29.5	<0.0001
balance	0.0055	0.0002	24.9	<0.0001

TABLE 4.1. *For the* Default *data, estimated coefficients of the logistic regression model that predicts the probability of* default *using* balance*. A one-unit increase in* balance *is associated with an increase in the log odds of* default *by* 0.0055 *units.*

- Prediction for Credit Card Stmt. balance = $1000 is 0.576 %

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1,000}}{1+e^{-10.6513+0.0055\times 1,000}} = 0.00576,$$

- Prediction for Credit Card Stmt. balance = $2000 is 58.6%

Predicting Defaulters!!

	Coefficient	Std. error	z-statistic	p-value
Intercept	−3.5041	0.0707	−49.55	<0.0001
student[Yes]	0.4049	0.1150	3.52	0.0004

TABLE 4.2. *For the* Default *data, estimated coefficients of the logistic regression model that predicts the probability of* default *using student status. Student status is encoded as a dummy variable, with a value of 1 for a student and a value of 0 for a non-student, and represented by the variable* student[Yes] *in the table.*

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=Yes}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431,$$

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=No}) = \frac{e^{-3.5041+0.4049\times 0}}{1+e^{-3.5041+0.4049\times 0}} = 0.0292.$$

Students tend to default more than non-students!

Multiple Logistic Regression

- $\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$
- Model with p predictors

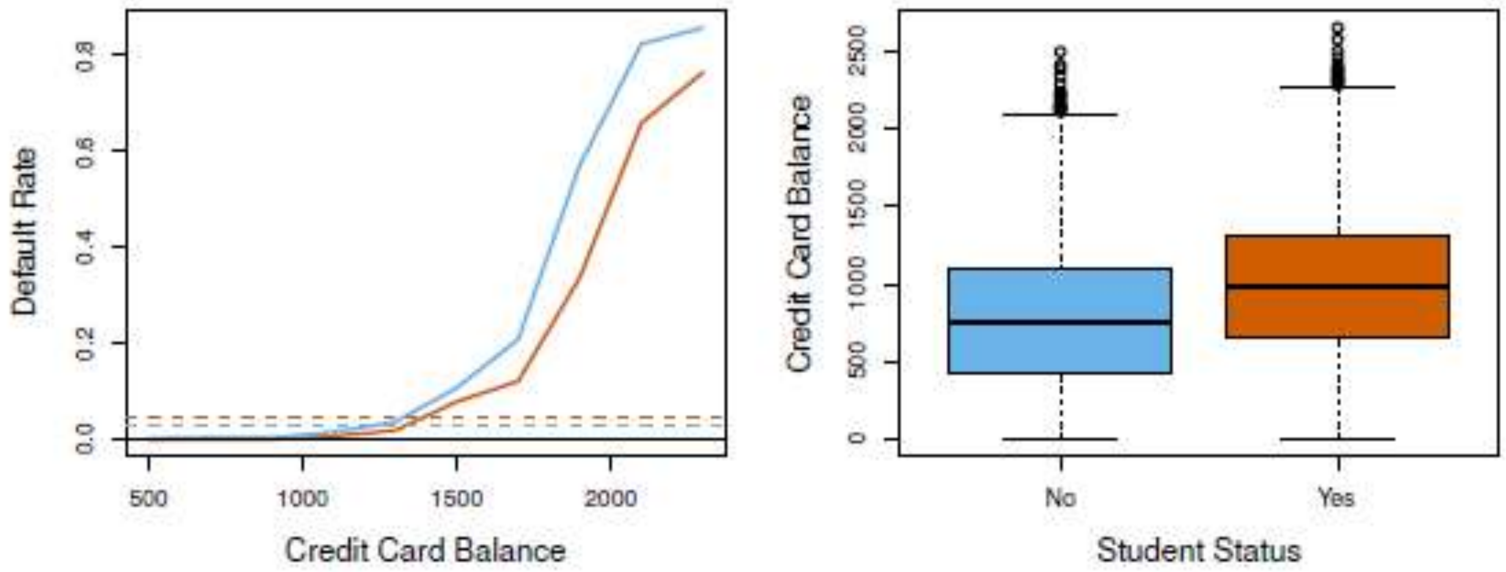
Predicting Defaulters!!

	Coefficient	Std. error	z-statistic	p-value
Intercept	−10.8690	0.4923	−22.08	<0.0001
balance	0.0057	0.0002	24.74	<0.0001
income	0.0030	0.0082	0.37	0.7115
student[Yes]	−0.6468	0.2362	−2.74	0.0062

TABLE 4.3. *For the* Default *data, estimated coefficients of the logistic regression model that predicts the probability of* default *using* balance, income, *and student status. Student status is encoded as a dummy variable* student[Yes], *with a value of 1 for a student and a value of 0 for a non-student. In fitting this model,* income *was measured in thousands of dollars.*

- Income: Is it significant predictor of defaulters?
- Coefficient for "Student=Yes" is –ve? How is it possible?

Solving the mystery (Confounding variable)



- The student default rate is at or below that of the non-student default rate for every value of balance.
- Default rates for students and non-students averaged over all values of balance and income, suggest the opposite effect: the overall student default rate is higher than the non-student default rate

- The variables student and balance are correlated. Students tend to hold higher levels of debt, which is associated with higher probability of default.
- A student is riskier than a non-student if no information about the student's credit card balance is available. However, that student is less risky than a non-student with the same credit card balance!

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 l Softmax function ilities are defined as

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

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 distri 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 $y_{n\ell'} = 0 \ \forall \ \ell' \neq \ell$

Multinomial Logistic Regression

- Softmax coding: K classes
  - $P[Y = k|X = x] = \frac{e^{w_{k0}+w_{k1}x_1+\cdots+w_{kp}x_p}}{\Sigma_{l=1}^{K} e^{w_{l0}+w_{l1}x_1+\cdots+w_{lp}x_p}}$
- We estimate the coefficients for all K classes.
- Log Odds:
  - $\log\left(\frac{P[Y = k|X = x]}{P[Y = k'|X = x]}\right) = (w_{k0} - w_{k'0}) + (w_{k1} - w_{k'1})x_1 + \cdots + (w_{kp} - w_{k'p})x_p$

Generative Models for Classification

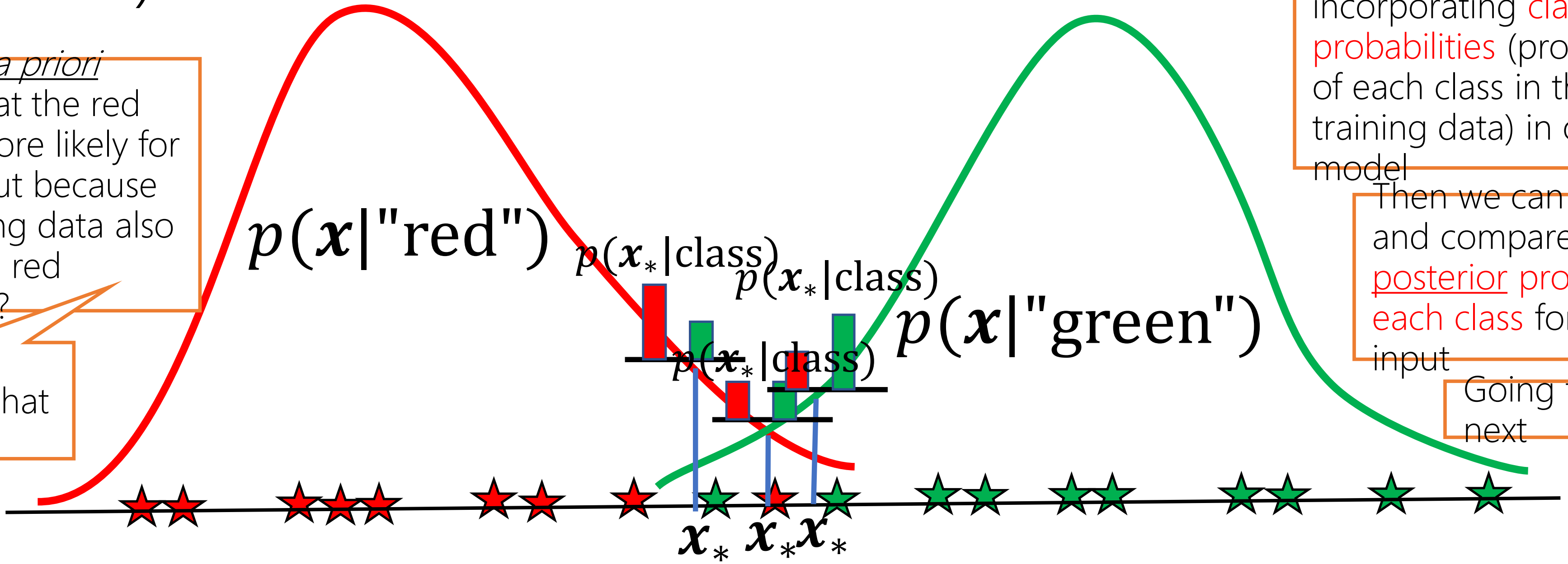
- Model the distribution of the predictors X separately in each of the response classes (for each value of Y)
  - Alternative to modeling p(Y|X)
  - We then use Bayes' theorem to flip these around into estimates for Pr(Y = k|X = x)
- When the distribution of X within each class is assumed to be normal, it turns out that the model is very similar in form to logistic regression.
- Why?
  - Logistic Regression is unstable when there is substantial separation between the two classes
  - Generative methods considered here do not suffer from this problem.
  - If the distribution of the predictors X is approximately normal in each of the classes and the sample size is small, then the approaches in this section may be more accurate than logistic regression.

Generative Classification: A Basic Idea

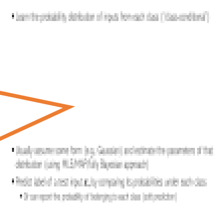
- Learn the probability distribution of inputs from each class ("class-conditional")

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

Generative Classification: More Generally..

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

- 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 $\boldsymbol{x}_*$

Class prior distribution for class k

Class-conditional distribution of inputs from class k

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

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

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

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

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

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. C… various factors

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

  - 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

- 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.

Gen. Classifn. using Gaussian Class-conditionals

- The generative classification model $p(y = k|\boldsymbol{x}) = \frac{p(y=k)p(\boldsymbol{x}|}{p(\boldsymbol{x}|\theta)}$

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

- Assume each class-conditional $p(\boldsymbol{x}|y = k)$ to be a Gaussian

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

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 $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$
  - Can estimate these using MLE/MAP/Bayesian inference
  - Already saw the MLE solution for $\boldsymbol{\pi}$: $\pi_k = N_k/N$ (can also do MA
  - MLE solution for $\boldsymbol{\mu}_k = \frac{1}{\ }\sum \ \boldsymbol{x}_n$, $\boldsymbol{\Sigma}_k = \frac{1}{\ }\sum \ (\boldsymbol{x}_n-\boldsymbol{\mu}_k)(\boldsymbol{x}_n-\boldsymbol{\mu}_k)^\top$

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

- st ... wi

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

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

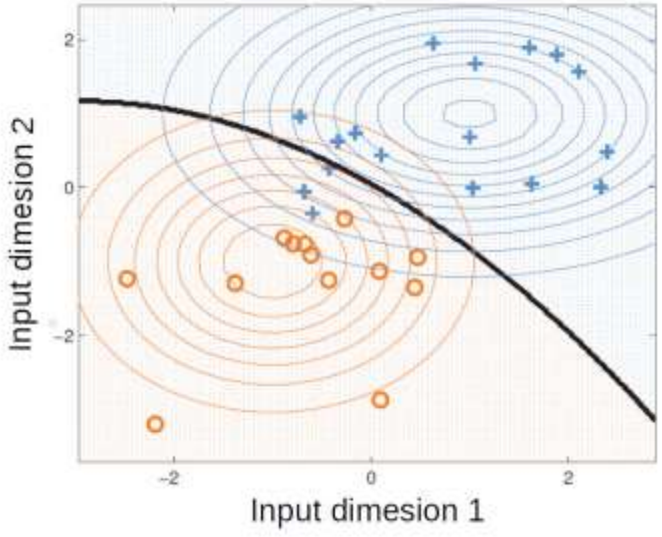
Note that the exponent has a Mahalanobis distance like term. Also, accounts for the fraction of training examples in class k

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 cont

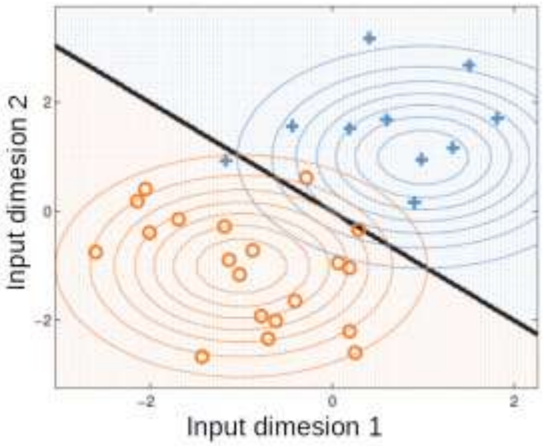
$$(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - (\mathbf{x}-\boldsymbol{\mu}_{k'})^\top \boldsymbol{\Sigma}_{k'}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{k'}) = 0$$

Decision boundary contains all inputs $\boldsymbol{x}$ that satisfy the above

This is a quadratic function of $\boldsymbol{x}$ (this model is sometimes

Decision Boundary with Gaussian Class-Conditional

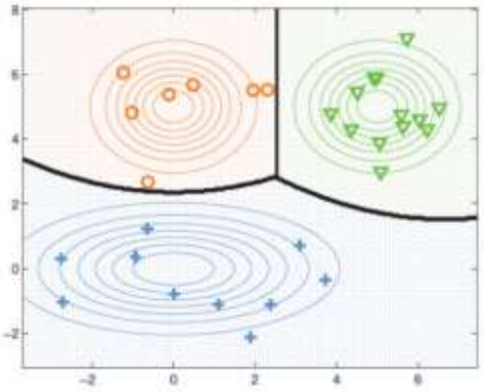
- Assume all classes are modeled using the same covariance matrix $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$
- In this case, the decision boundary b/w any pair of classes will be linear

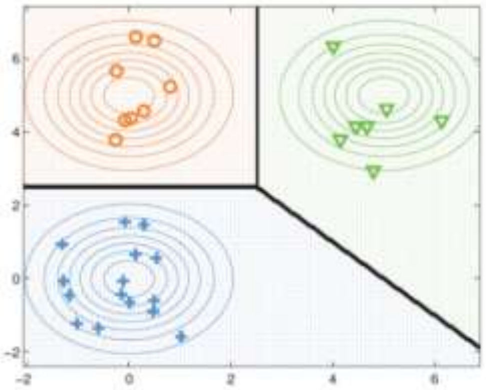


Reason: Again using $p(y = k|x, \theta) = p(y = k'|x, \theta)$, comparing their logs and ignoring terms that don't contain $\boldsymbol{x}$, we have

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

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

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

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

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

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

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 <u>exactly</u> 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 condition

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

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 f nditionals $p(\boldsymbol{x}|y=k)$ 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 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

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 $p(y|x)$
- Generative approaches model $p(y|x)$ via $p(x, y)$
- 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

Discriminative vs Generative (Contd)

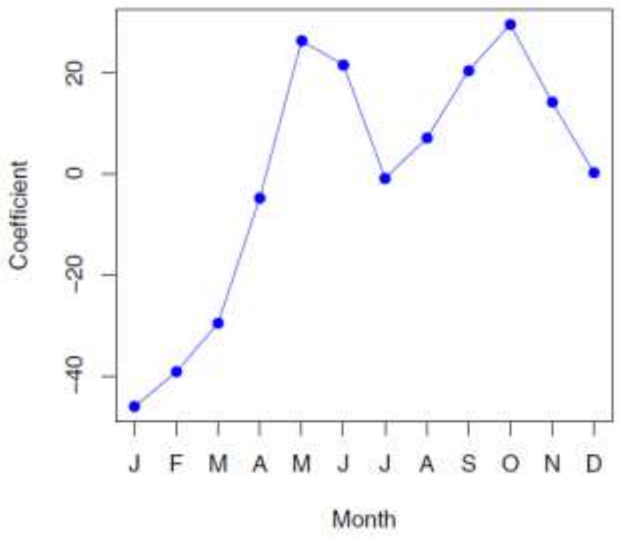
- Leveraging unlabeled data: Generative models can handle this easily by treating the missing labels are latent variables and are ideal for Semi-supervised Learning. Discriminative models can't do it easily
- Adding data from new classes: Discriminative model will need to be re-trained on all classes 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)

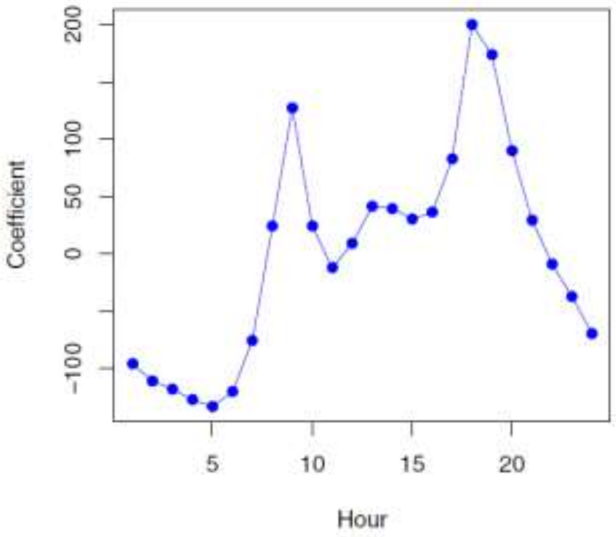
Generalized Linear Models

- We may sometimes be faced with situations in which Y is neither qualitative nor quantitative, and so neither linear regression or classification approaches covered so far are applicable.
- Example of Bikeshare dataset
- Response: bikers, the number of hourly users of a bike sharing program in Washington, DC. This response value is neither qualitative nor quantitative: instead, it takes on non-negative integer values, or counts
- Covariates:
    - mnth (month of the year),
    - hr (hour of the day, from 0 to 23),
    - workingday (an indicator variable that equals 1 if it is neither a weekend nor a holiday),
    - temp (the normalized temperature,in Celsius), and
    - weathersit : {clear; misty or cloudy; light rain or light snow; or heavy rain or heavy snow}

Linear regression model

	Coefficient	Std. error	z-statistic	p-value
`Intercept`	73.60	5.13	14.34	0.00
`workingday`	1.27	1.78	0.71	0.48
`temp`	157.21	10.26	15.32	0.00
`weathersit[cloudy/misty]`	-12.89	1.96	-6.56	0.00
`weathersit[light rain/snow]`	-66.49	2.97	-22.43	0.00
`weathersit[heavy rain/snow]`	-109.75	76.67	-1.43	0.15



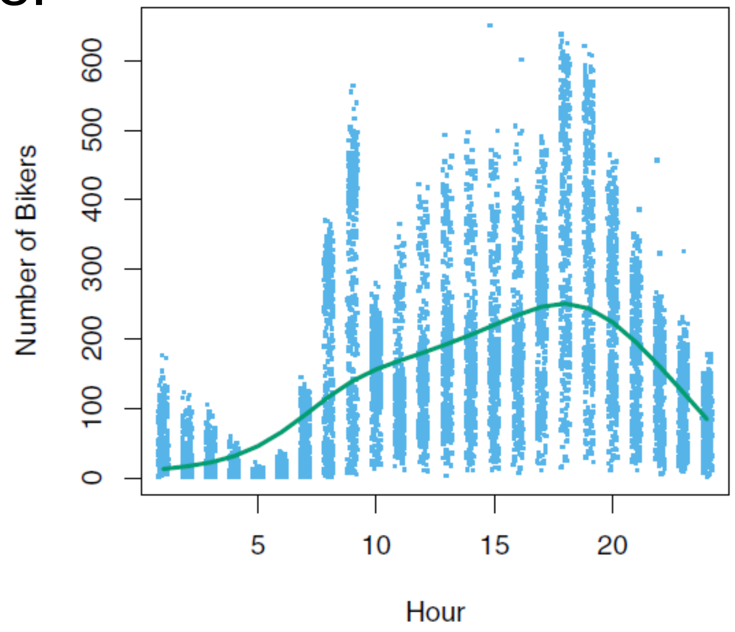


Month
Hr
Weathersit
Are considered qualitative variables

What could be the problems with this?

- 9.6% of the fitted values in the Bikeshare data set are negative: that is, the linear regression model predicts a negative number of users during 9.6% of the hours in the data set.
    - Can we perform meaningful predictions on the data?
    - Concerns about the accuracy of the coefficient estimates, confidence intervals, and other outputs of the regression model
- The number of bikers is +ve, integer valued

Heteroskedastic: As the mean number of bikers increase, so does the variance

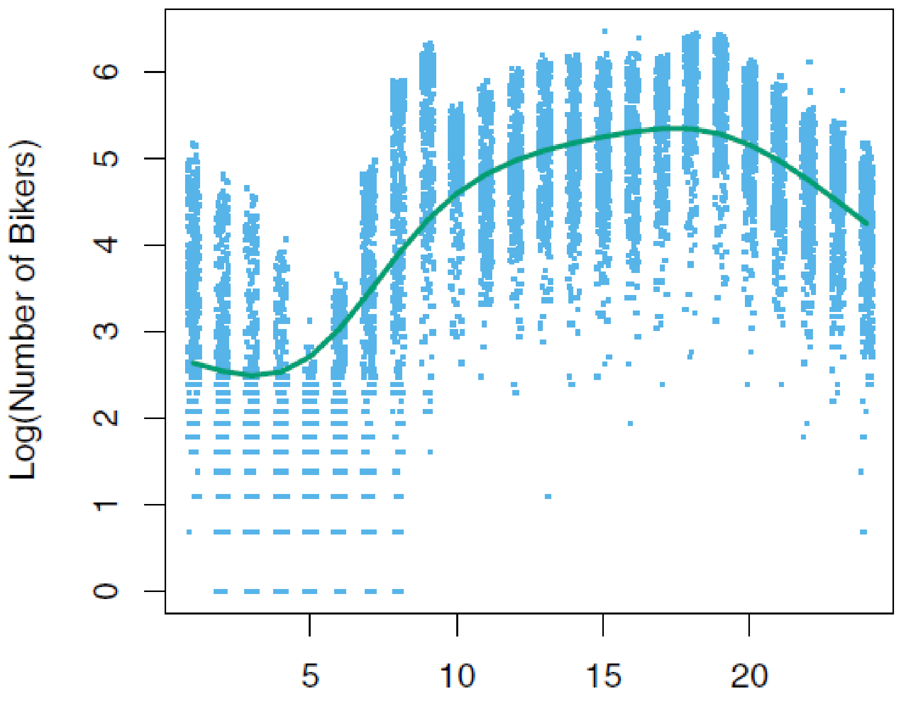


Possible solution

- Log transformation of the response, Y

$$\log(Y) = \Sigma_{j=1}^{p} \mathrm{X_j w_j} + \epsilon$$

- Avoids the possibility of negative predictions,
- Overcomes much of the heteroscedasticity in the untransformed data



Poisson Regression

- Poisson Distribution: $P[Y = k] = \frac{e^{-\lambda}\lambda^k}{k!}$ for k = 0,1,2…
- λ>0, is the expected value of Y = Var[Y]
- We expect the mean number of users of the bike sharing program, λ = E(Y), to vary as a function of the hour of the day, the month of the year, the weather conditions, and so forth.
- So rather than modeling the number of bikers, Y , as a Poisson distribution with a fixed mean value like λ = 5, we would like to allow the mean to vary as a function of the covariates.
  - $\log(\lambda) = w_0 + w_1 x_1 + \cdots + w_p x_p$

Results of Poisson Regression

	Coefficient	Std. error	z-statistic	p-value
`Intercept`	4.12	0.01	683.96	0.00
`workingday`	0.01	0.00	7.5	0.00
`temp`	0.79	0.01	68.43	0.00
`weathersit[cloudy/misty]`	-0.08	0.00	-34.53	0.00
`weathersit[light rain/snow]`	-0.58	0.00	-141.91	0.00
`weathersit[heavy rain/snow]`	-0.93	0.17	-5.55	0.00

- Parameters are estimated using MLE approach
- $l(w_0, w_1, \ldots, w_p) = \Pi_{i=1}^{n} \frac{e^{-\lambda(x_i)} \lambda(x_i)^{y_i}}{y_i!}$
- Where $\log(\lambda(x_i)) = w_0 + w_1 x_1 + \cdots + w_p x_p$

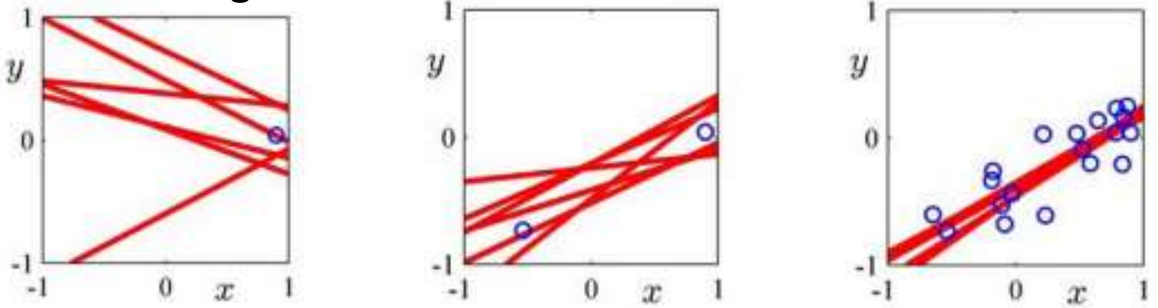
Notable Points

Generalization

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 o c 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)