Probabilistic Models for Supervised Learning (contd)

CS771: Introduction to Machine Learning

Prob. Models for Supervised Learning

Non-probabilistic supervised learning approaches (e.g., SVM) are usually considered discriminative since $p(\boldsymbol{x})$ is never modeled

- Goal: Learn the conditional distribution $p(y|\boldsymbol{x})$. Broadly, two approaches

Discriminative Approach

$$p(y|\boldsymbol{x}) = p(y|f(\boldsymbol{x}, \boldsymbol{w}))$$

f can be any function which uses inputs and weights $\boldsymbol{w}$ to defines parameters of distr. p

Some examples

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

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

Generative Approach

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

Requires estimating the joint distribution of inputs and outputs to get the conditional $p(y|\boldsymbol{x})$ (unlike the discriminative approach which directly estimates the conditional $p(y|\boldsymbol{x})$ and does not model the distribution of $\boldsymbol{x}$)

- Both approaches have their pros and cons (discussed later)

The Discriminative Approach

- This approach models $\boldsymbol{p(y|x)}$ <u>directly</u> using a suitable prob. distribution, e.g.,

Regression model likelihood

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

Binary classification model likelihood

$\mu_i = \sigma(\boldsymbol{w}^\top\boldsymbol{x}_i)$

$$p(y_i|\boldsymbol{x}_i,\boldsymbol{w}) = \text{Bernoulli}(y_i|\sigma(\boldsymbol{w}^\top\boldsymbol{x}_i)) = \mu_i^{y_i}(1-\mu_i)^{1-y_i}$$

- The negative log-likelihood (assuming i.i.d. outputs) for the above two models

$$NLL(\boldsymbol{w}) = \sum_{i=1}^{N} -\log p(y_i|\boldsymbol{x}_i,\boldsymbol{w}) = \sum_{i=1}^{N} -\log\sqrt{\frac{\beta}{2\pi}}\exp\left[-\frac{\beta}{2}(y_i-\boldsymbol{w}^\top\boldsymbol{x}_i)^2\right]$$

$$NLL(\boldsymbol{w}) = \frac{\beta}{2}\sum_{n=1}^{N}(y_i-\boldsymbol{w}^\top\boldsymbol{x}_i)^2$$ (same as squared loss ☺)

$$NLL(\boldsymbol{w}) = \sum_{i=1}^{N} -\log p(y_i|\boldsymbol{x}_i,\boldsymbol{w}) = \sum_{i=1}^{N} -\log \mu_i^{y_i}(1-\mu_i)^{1-y_i}$$

$$NLL(\boldsymbol{w}) = -\sum_{n=1}^{N}[y_i\log\mu_i + (1-y_i)\log(1-\mu_i)]$$ (same as cross-entropy loss ☺)

Thus minimization of NLL (i.e., doing MLE) is equivalent to minimization of the training loss

Lacks regularization (and thus can overfit) but we can use a prior on $\boldsymbol{w}$ to do MAP estimation

A prior over $\boldsymbol{w}$

- For MAP estimation (and to compute full posterior), we need a prior $\boldsymbol{w} \in \mathbb{R}^D$
- 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 $\boldsymbol{w}_0$

Σ specifies how strong our belief is that $\boldsymbol{w}$ to close to $\boldsymbol{w}_0$

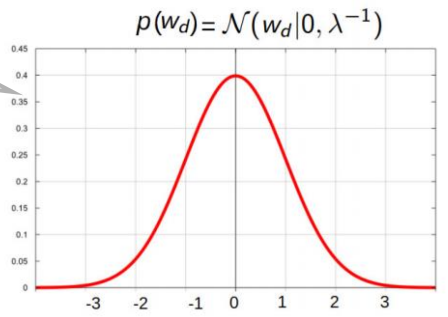
- A specific example of a multivariate Gaussian prior in this problem

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

Omitting λ for brevity

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

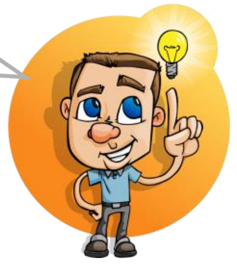
The precision λ of the Gaussian prior controls how aggressively the prior pushes the elements towards mean (0)



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

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

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



That's indeed the case ☺

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

MAP Estimation with $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|0, \lambda^{-1}\boldsymbol{I}_D)$

- The MAP objective (log-posterior) will be the log-likelihood + $\log p(\boldsymbol{w})$
- Ignoring terms that don't depend on $\boldsymbol{w}$ the log-posterior will be

$$NLL(\boldsymbol{w}) - \log \exp\left[-\frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w}\right] = NLL(\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w}$$

- Exercise: For regression with likelihood $p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \mathcal{N}(y_i|\boldsymbol{w}^\top\boldsymbol{x}_i, \beta^{-1})$

$$\hat{w}_{MAP} = \operatorname*{argmin}_{\boldsymbol{w}} \frac{\beta}{2}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^\top\boldsymbol{x}_i)^2 + \frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w} = (\boldsymbol{X}^\top\boldsymbol{X} + \frac{\lambda}{\beta}\boldsymbol{I}_D)^{-1}\ \boldsymbol{X}^\top\boldsymbol{y}$$

- Classification with likelihood $p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \text{Bernoulli}(y_i|\sigma(\boldsymbol{w}^\top\boldsymbol{x}_i))$ is the same as logistic regression. When also using the above prior, the MAP solution will be the same as cross-entropy loss minimization + L2 regularization on the weights

Prob. Linear Regression: The Full Posterior

- If we want the full posterior distribution over $\boldsymbol{w}$, it can be computed as

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

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

- For regression, with Gaussian likelihood and zero mean Gaussian prior are conjugate (both Gaussians), and the posterior will be Gaussian

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

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

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

Will provide the proof separately but it is straightforward when using properties of Gaussian

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)

- For binary classification with Bernoulli likelihood and zero mean Gaussian prior, we don't have conjugacy, and the posterior can't be computed in closed form

Logistic Regression: The Full Posterior

- The posterior distribution for the logistic regression model

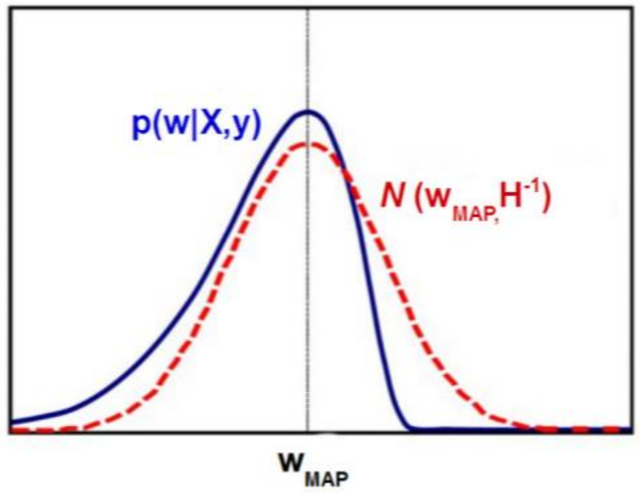
Gaussian

Bernoulli

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

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

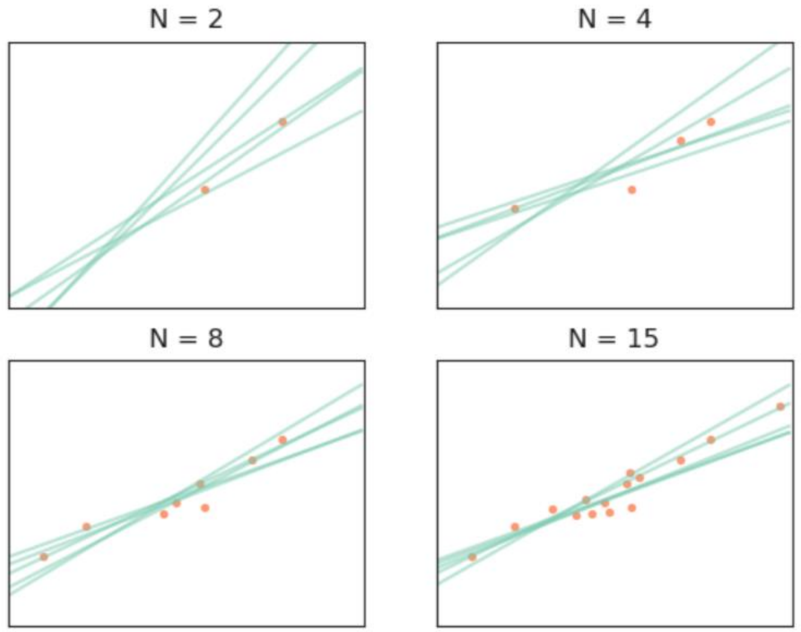


Approximates the posterior of $\boldsymbol{w}$ by a Gaussian whose mean is the MAP solution $\hat{\boldsymbol{w}}_{MAP}$ and covariance matrix is the inverse of the Hessian (Hessian: second derivative of the negative log-posterior of the LR model)

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

What does posterior of a linear model look like ?

- 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)

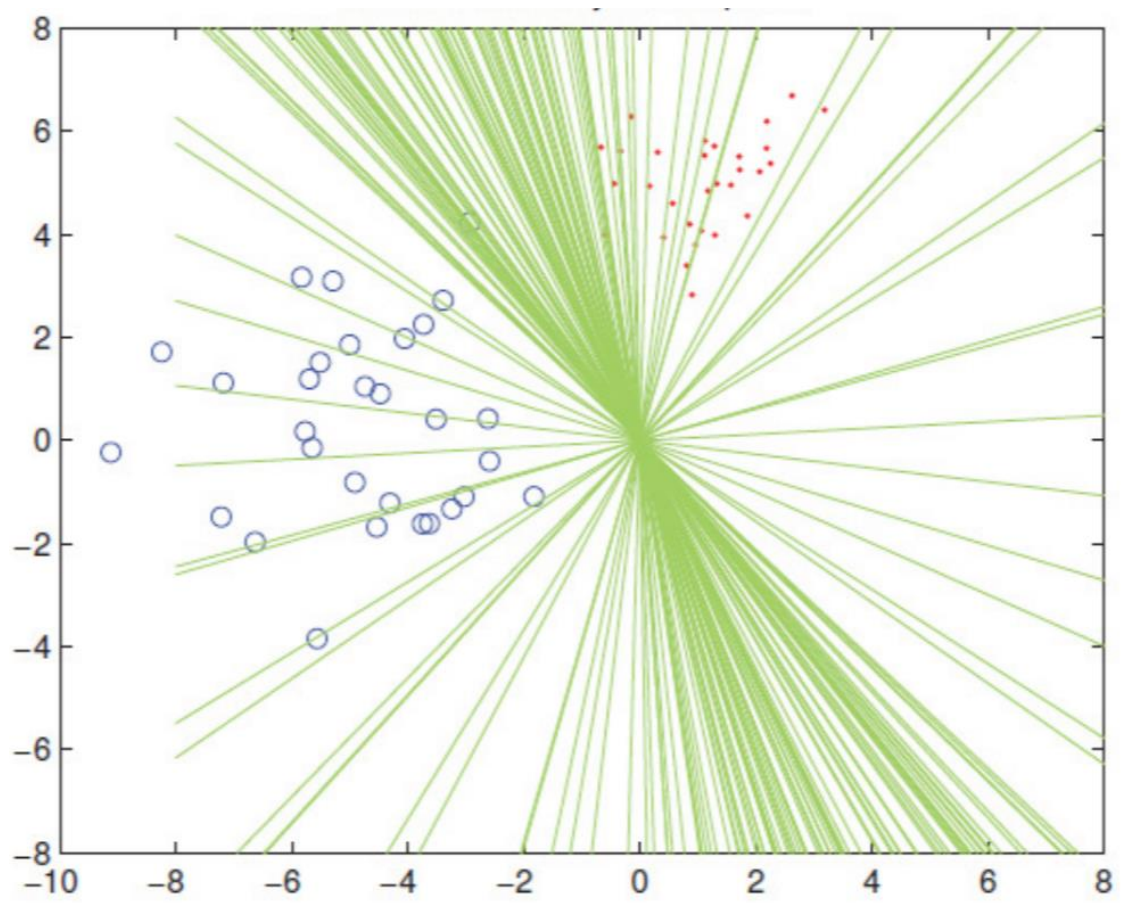
Importance of each solution in this ensemble is its posterior probability $p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y})$

- 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 N increases (we become more sure when we see more training data)

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

What does posterior of a linear model look like ?

- Can also sample from the Laplace approximate posterior of logistic regression
- 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

Prob. Linear Regression: 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

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

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

- If we have the full posterior over $\boldsymbol{w}$, can compute the posterior predictive dist.

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

Assuming the hyperparameters λ and β are known (otherwise, the PPD can't be computed exactly)

- Requires an integral but has a closed form

Mean prediction

Will provide the proof separately but it is straightforward when using properties of Gaussian

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

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)

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 $p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) \approx \frac{1}{M}\sum_{m=1}^{M} p(y_* = 1|\boldsymbol{w}_\text{m}, \boldsymbol{x}_*) = \frac{1}{M}\sum_{m=1}^{M} \sigma(\boldsymbol{w}_m^\top \boldsymbol{x}_n)$

Recap: Prob. Models for Supervised Learning

- Goal: Learn the conditional distribution $\boldsymbol{p(y|x)}$. Broadly, two approaches

Discriminative Approach

$$p(y|\boldsymbol{x}) = p(y|f(\boldsymbol{x},\boldsymbol{w}))$$

f can be any function which uses inputs and weights $\boldsymbol{w}$ to defines parameters of distr. $\boldsymbol{p}$

Some examples

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

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

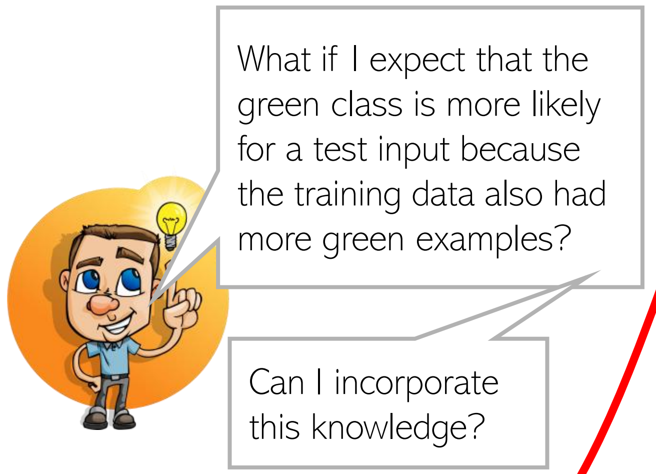
Generative Approach

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

Requires estimating the joint distribution of inputs and outputs to get the conditional $p(y|\boldsymbol{x})$ (unlike the discriminative approach which directly estimates the conditional $p(y|\boldsymbol{x})$ and does not model the distribution of $\boldsymbol{x}$)

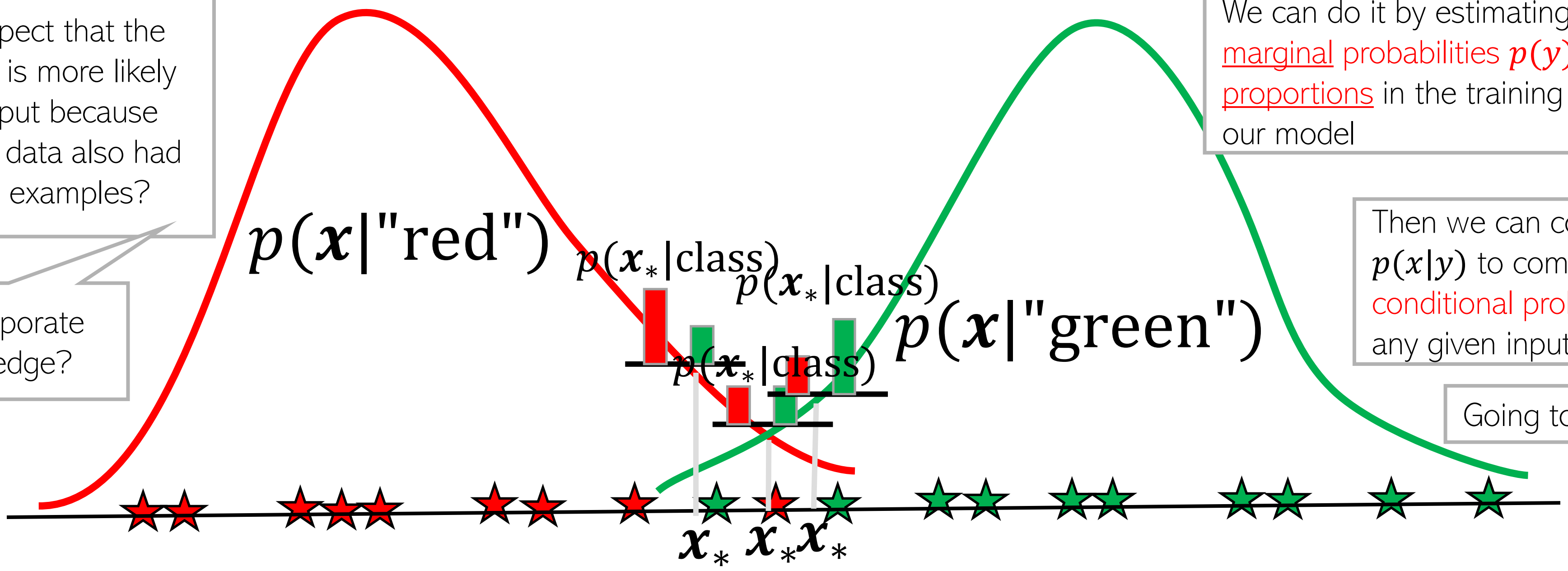
Generative Classification: A Basic Idea

- Learn the probability distribution $p(x|y=k)$ of inputs from each class k



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

Can I incorporate this knowledge?


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

Yes. Possible with generative model. We can do it by estimating class marginal probabilities $p(y)$ (class proportions in the training data) in our model

Then we can combine $p(y)$ and $p(x|y)$ to compute $p(y|x)$ - conditional probability of label for any given input

Going to talk about this next

- We usually assume some form for $p(x|y=k)$ (e.g., Gaussian) and estimate the parameters of that distribution (MLE/MAP/fully posterior)
- We then predict label of test input $\boldsymbol{x}_*$ by comparing probabilities under each class
  - Or can report the probability of belonging to each class (soft prediction)

Generative Classification

- Suppose we have training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ from K classes
- The conditional probability of label y_n given the input $\boldsymbol{x}_n$

$$p(y_n = k|x_n) = \frac{p(\boldsymbol{x}_n, y_n = k)}{p(x_n)}$$

$$= \frac{p(y_n = k) \times p(\boldsymbol{x}_n | y_n = k)}{p(\boldsymbol{x}_n)}$$

Known as the "class-conditional" distribution

Probability distribution of the inputs from class $\boldsymbol{k}$

The numerator (joint distribution of x_n and y_n) summed over all K values of y_n

Known as "class-marginal" or "class-prior" distribution

Marginal distribution of just the labels (not looking at the inputs) – Bernoulli/multinoulli

Marginal distribution of the input $\boldsymbol{x}_n$

- We use the training data to estimate the class-marginal and class-conditionals

Estimating Class Marginals

- Estimating class marginals $p(y = k)$ is usually straightforward
- Since labels are discrete, we assume class marginal $p(y)$ to be a multinoulli

If only two classes, assume Bernoulli

$\pi_k = p(y = k)$

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

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

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

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

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

- MLE solution is $p(y = k) = \pi_k = N_k/N$ where $N_k = \sum_{n=1}^{N} \mathbb{I}[y = k]$
  - Thus $p(y = k) = \pi_k$ is simply the fraction of inputs from class k
- Can also compute MAP estimate or full posterior of $\boldsymbol{\pi}$ using a Dirichlet prior

Estimating Class-Conditionals

To be estimated using the N_k training inputs $\{\boldsymbol{x}_n : y_n = k\}$ from class k

- Can assume a distribution $p(\boldsymbol{x}|y=k) = p(\boldsymbol{x}|\theta_k)$ for inputs of each class k
- If $\boldsymbol{x}$ is D-dimensional, $p(\boldsymbol{x}|\theta_k)$ will be a D-dimensional distribution
- Can compute MLE/MAP estimate or full posterior of θ_k
  - This essentially is a density estimation problem for the class-cond.
  - In principle, can use any density estimation method

E.g., if $p(\boldsymbol{x}|\theta_k)$ is multivariate Gaussian then assume it to have a diagonal covariance matrix instead of full covariance matrix

Such assumptions greatly reduce the number of parameters to be estimated

- Choice of the form of $p(\boldsymbol{x}|\theta_k)$ 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 other more sophisticated distributions
  - Amount of training data available (important)
    - If D large and N_k small, it will be difficult to get a good estimate θ_k

In such cases, we may need to regularize θ_k or make some simplifying assumptions on $p(\boldsymbol{x}|\theta_k)$, such as features being conditionally independent given class e.g., $p(\boldsymbol{x}|\theta_k) = \prod_{d=1}^{D} p(x_d|\theta_{kd})$ - naïve Bayes

Especially if the number of features (D) is very large because large value of D means k consists of a large number of parameters (e.g., in the Gaussian case, $\theta_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, D params for $\boldsymbol{\mu}_k$ and $O(D^2)$ params for $\boldsymbol{\Sigma}_k$. Can overfit

Generative Classification: At Test Time

- Recall the form of the conditional distribution of the label

Class-marginal accounts for the frequency of class k labels in the training data

Class-conditional distribution of inputs accounts for the shape/spread of class k

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

$$\propto p(y_* = k) \times p(\boldsymbol{x}_*|y_* = k)$$

$$\propto \hat{\pi}_k \times p(\boldsymbol{x}_*|\hat{\theta}_k)$$

Probability of $\boldsymbol{x}_*$ belonging to class k is proportional to the fraction of training inputs from class k times the probability of $\boldsymbol{x}_*$ under the distribution of inputs from class k

- If we assume the class-marginal to be uniform $(p(y_* = k) = 1/K)$ then

$$p(y_* = k|\boldsymbol{x}_*) \propto p(\boldsymbol{x}_*|\hat{\theta}_k)$$

- The most likely label is $y_* = \text{argmax}_{k\in\{1,2,\ldots,K\}}\, p(y_* = k|\boldsymbol{x}_*)$

Gen. Classifn. using Gaussian Class-conditionals

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

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 marginal is multinoulli $p(y=k) = \pi_k, \pi_k \in (0,1), \sum_{k=1}^{K}\pi_k = 1$
- 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
  - The MLE solution for $\boldsymbol{\pi}$: $\pi_k = N_k/N$ (exercise)
  - 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

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

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

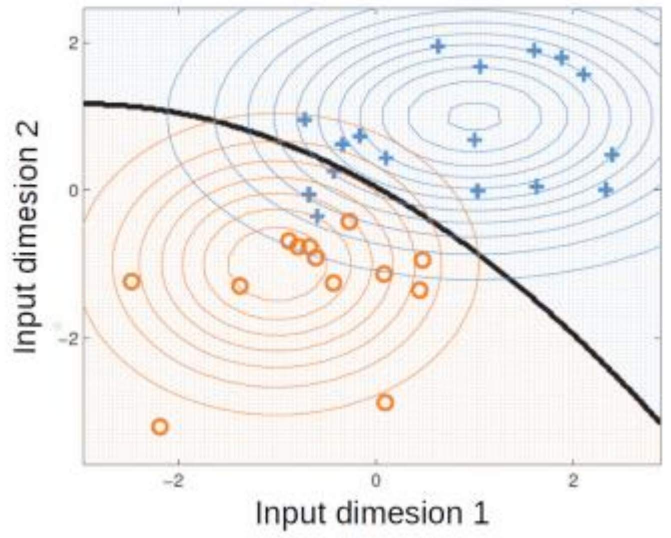
Note that the exponent has a Mahalanobis distance like term. Also, accounts for the fraction of training examples in class k

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



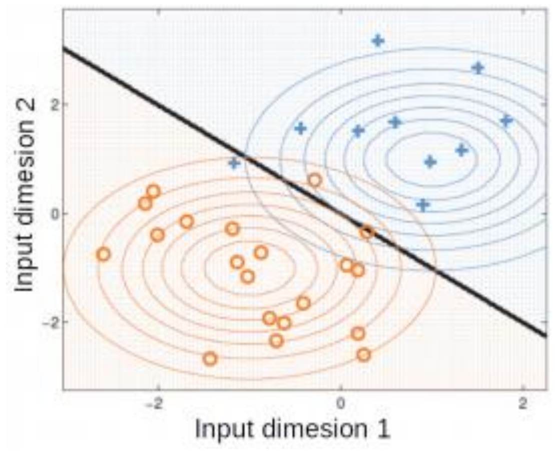
Reason: For any two classes k and k' at the decision boundary, we will have $p(y = k|x, \theta) = p(y = k'|x, \theta)$. Comparing their logs and ignoring terms that don't contain $\boldsymbol{x}$, can easily see that

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

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

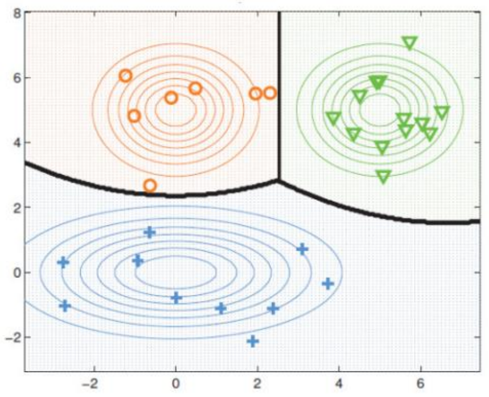
- Assume all classes are modeled using the same covariance matrix $\mathbf{\Sigma}_k = \mathbf{\Sigma}, \forall k$
- In this case, the decision boundary b/w any pair of classes will be linear

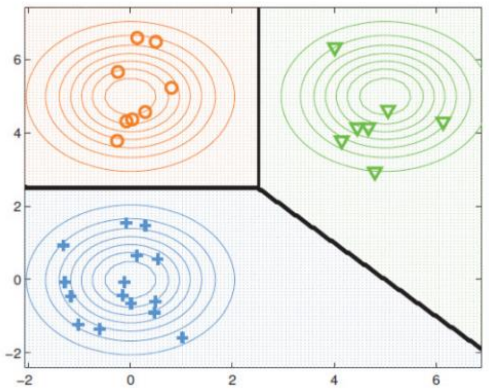


Reason: Again using $p(y = k|x, \theta) = p(y = k'|x, \theta)$, comparing their logs and ignoring terms that don't contain $\boldsymbol{x}$, we have

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

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 $\mathbf{\Sigma}$)





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 conditional 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

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

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

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

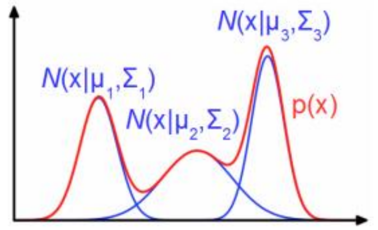
$$\hat{y} = \arg\min_k \quad (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

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 exactly the LwP rule
  - Thus LwP assumes spherical classes with roughly equal number of inputs from each class

Unsupervised Generative Classification

- In generative classification, we estimate
  - Class marginal distribution $p(y) = \text{multinoulli}(y|\boldsymbol{\pi})$
  - K class-conditional distributions $p(x|\theta_k), k = 1,2,\ldots,K$



- Can we estimate $\{\pi, \theta_1, \theta_2, \ldots, \theta_K\}$ if the training labels are not known?

- It then becomes an unsupervised learning problem
  - Mixture modeling or clustering

- We will look at it later but the general idea is based on ALT-OPT
  1. Guess the label of each input given current estimate of $\{\pi, \theta_1, \theta_2, \ldots, \theta_K\}$
  2. Re-estimate $\{\pi, \theta_1, \theta_2, \ldots, \theta_K\}$ given the label guesses
  3. Alternate between steps 1 and 2 till convergence

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 $\boldsymbol{x}$ by learning the class-conditional

- Recall that discriminative approaches model $p(\boldsymbol{y}|\boldsymbol{x})$ directly
- Generative approaches model $p(\boldsymbol{y}|\boldsymbol{x})$ via $p(\boldsymbol{x},\boldsymbol{y})$
- 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 (will see later)
- Inputs with features having mixed types: Generative model can handle this
  - Appropriate $p(x_d|y)$ 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)