CSDS 440: Machine Learning

Soumya Ray (he/him, sray@case.edu)

Olin 516

Office hours T, Th 11:15-11:45 or by appointment

Why does Naïve Bayes work well?

- Very simplistic independence assumptions
  - Everyone knows that these assumptions are nearly always wrong
  - But, paradoxically, often works well in practice

- Why?
  - Works well for *classification*, but not so great at *density estimation*
  - Most probabilities end up near 0/1 (ask for paper)

Logistic Regression

- Simplest Discriminative model
- Models log odds as a linear function

$$\log \frac{p(Y=1|\mathbf{x})}{p(Y=-1|\mathbf{x})} = \mathbf{w}\bullet\mathbf{x} + b$$

$$p(Y=1|\mathbf{x}) = \left[1 - p(Y=1|\mathbf{x})\right] e^{(\mathbf{w}\bullet\mathbf{x}+b)}$$

$$p(Y=1|\mathbf{x})(1+e^{(\mathbf{w}\bullet\mathbf{x}+b)}) = e^{(\mathbf{w}\bullet\mathbf{x}+b)}$$

$$p(Y=1|\mathbf{x}) = \frac{e^{(\mathbf{w}\bullet\mathbf{x}+b)}}{1+e^{(\mathbf{w}\bullet\mathbf{x}+b)}} = \frac{1}{1+e^{-(\mathbf{w}\bullet\mathbf{x}+b)}}$$

Classification with LR

- LR directly specifies $p(Y=1|\mathbf{x})$, compute and check if greater than 1/2

Estimating parameters

- Use MLE, optimize *conditional* log likelihood of the data

$$\mathbf{w}, b = \arg\max \prod_i p(Y_i = y_i \mid \mathbf{x}_i)$$

Conditional Likelihood

Conditional Log Likelihood

$$= \arg\max \sum_{i \in pos} \log p(Y_i = 1 \mid \mathbf{x}_i) + \sum_{i \in neg} \log p(Y_i = -1 \mid \mathbf{x}_i)$$

$$= \arg\max \sum_{i \in pos} \log\left(\frac{1}{1+e^{-\mathbf{w}\bullet\mathbf{x}+b}}\right) + \sum_{i \in neg} \log\left(1 - \frac{1}{1+e^{-\mathbf{w}\bullet\mathbf{x}+b}}\right)$$

Overfitting control

- Can include a term for overfitting control:

$$\mathbf{w}, b = \arg\max \sum_{i \in pos} \log\left(\frac{1}{1+e^{-\mathbf{w}\bullet\mathbf{x}+b}}\right) + \sum_{i \in neg} \log\left(1 - \frac{1}{1+e^{-\mathbf{w}\bullet\mathbf{x}+b}}\right)$$

$$\mathbf{w}, b = \arg\min \frac{1}{2}\left\|\mathbf{w}\right\|^2 + C\left[\begin{array}{l} \displaystyle\sum_{i \in pos} -\log\left(\frac{1}{1+e^{-\mathbf{w}\bullet\mathbf{x}+b}}\right) \\ \displaystyle + \sum_{i \in neg} -\log\left(1 - \frac{1}{1+e^{-\mathbf{w}\bullet\mathbf{x}+b}}\right) \end{array}\right]$$

Negative Conditional Log Likelihood

Estimating parameters

- Can use gradient descent, Newton methods etc

- Very robust method, works extremely well in many practical situations, very easy to code

Logistic Regression Geometry

- Classify as positive iff:

$$\frac{p(Y=1\,|\,\mathbf{x})}{p(Y=-1\,|\,\mathbf{x})} > 1$$

$$\text{or if } \log\frac{p(Y=1\,|\,\mathbf{x})}{p(Y=-1\,|\,\mathbf{x})} > 0$$

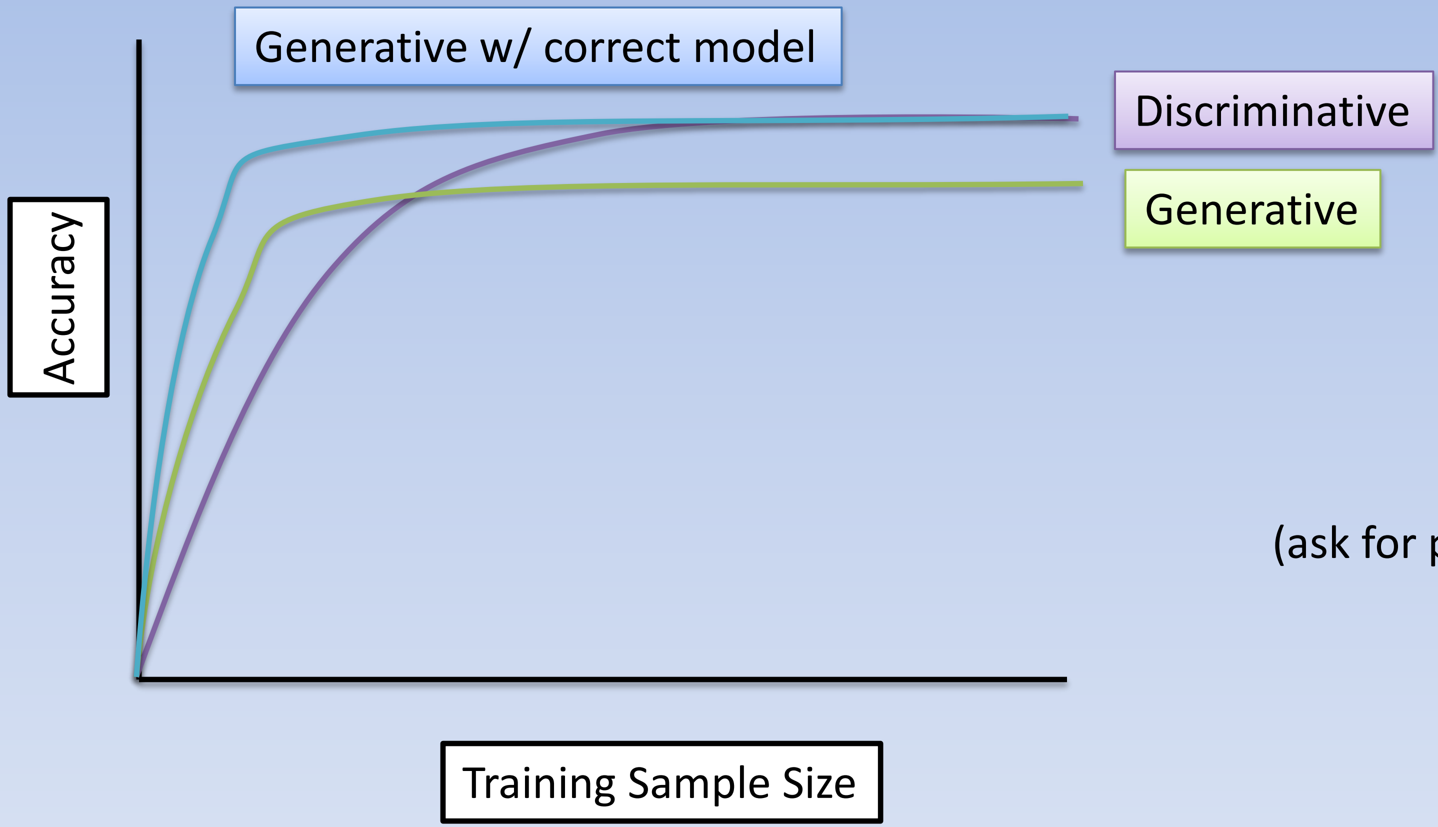
$$\text{But } \log\frac{p(Y=1\,|\,\mathbf{x})}{p(Y=-1\,|\,\mathbf{x})} = \mathbf{w}\bullet\mathbf{x} + b$$

So classify as positive iff $\mathbf{w}\bullet\mathbf{x} + b > 0$

Relationship to Naïve Bayes

- LR does not make the independence assumptions of NB
  - Can be more robust than NB, especially in the presence of irrelevant attributes
  - Also handles continuous attributes nicely
  - But (as with all discriminative models) no easy way to handle data issues such as missing data

Generative and Discriminative Pairs



(ask for paper)

Pros and Cons of Probabilistic Classification

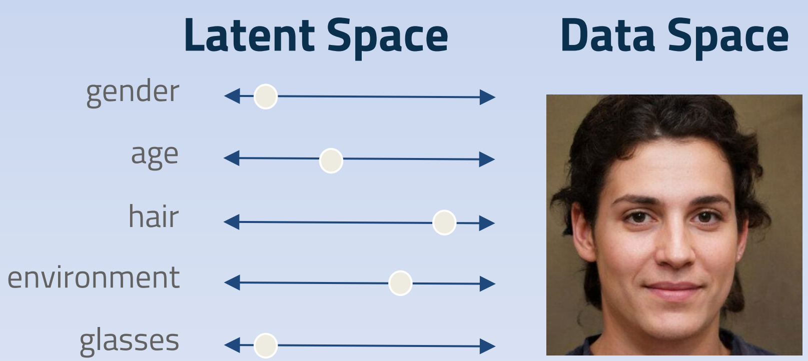
+ Optimal approach in decision-theoretic terms
+ Can incorporate prior knowledge (possibly later)
+ Produces confidence measures
+ Very well studied
+ Simple models are easy to implement
+ Can nicely capture causal influences (CSDS 442)
- Inference and estimation are in general hard
- Discriminative approaches can be hard to interpret

High dimensional Generative Models

- Suppose we wish to generate an image of a face
- This is hard!
- These are samples from a VERY high dimensional distribution
- And, the “axes” are not independent

Latent Variable Models

- To make the problem easier, we introduce "latent variables" z
  - These are hidden features which capture (hopefully independent) abstractions that constrain the space of images



Maximizing Likelihood

- Observe

$$p(X) = \int p(X \mid z)\, p(z)\, dz$$

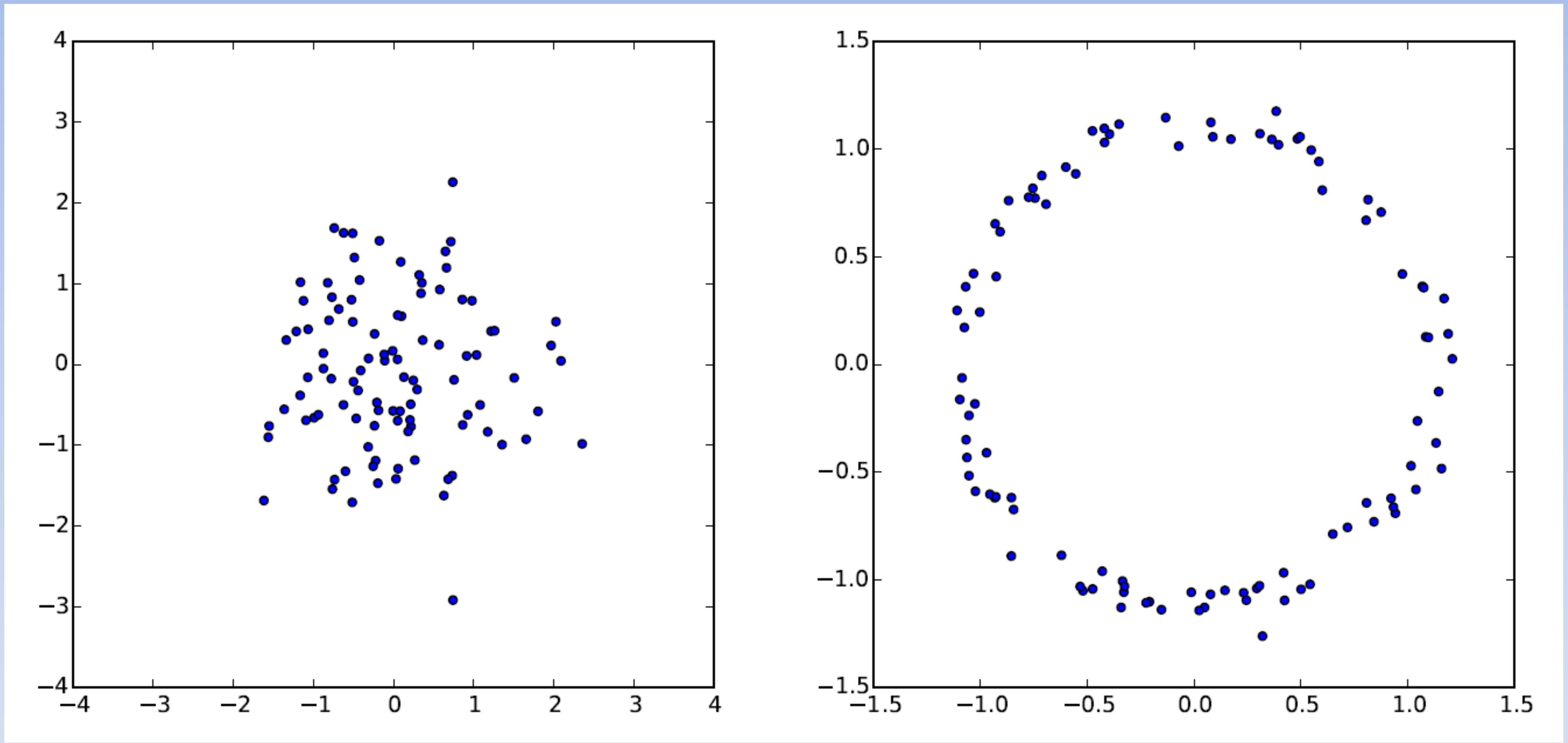
- To train a model, want to maximize LHS as before

1. What should z be?
2. How to compute the $p(X)$ above?

First clever idea

- We have no idea what z could be
- Let us just sample z from $N(\mathbf{0}, \boldsymbol{I})$ and use a nonlinear function to *transform* this input into the output we need
  - Does such a function exist?
  - In many cases yes! under "compatibility" conditions for $p(X)$ and sufficiently rich nonlinear transformations

You’re Joking, Right?



Left: samples z from 2D $N(0,I)$. Right: $f(z)=z/10+z/\|z\|$

And so

- We'll choose a trainable family $f_\theta(z)$, typically a neural network

- With this choice, $p(X \mid z) = N(f_\theta(z), \sigma^2 I)$

Evaluating Likelihood

$$p(X) = \int p(X \mid z) p(z) dz, z \sim N(0, I)$$

$$\approx \frac{1}{n} \sum_i p(X \mid z_i)$$

- Unfortunately, in high dimensions, most $p(X \mid z_i)$ will be near zero, so this is going to be VERY inefficient

Second key idea

- What if we had a function $Q(z \mid X)$, that could return a distribution over those z's that are likely to produce X?

- Then maybe we could use $E_{z \sim Q}\, p(X \mid z)$ to get a good approximation to the likelihood?

Aside: Kullback-Liebler divergence

- One way to measure the “dissimilarity” between two distributions

$$D(X(z)\,\|\,Y(z)) = E_{z\sim X}\left(\log\left(X(z)\right) - \log\left(Y(z)\right)\right)$$