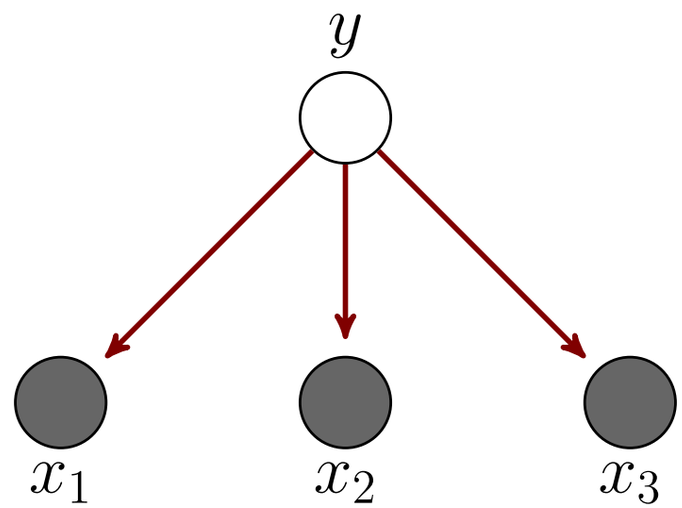
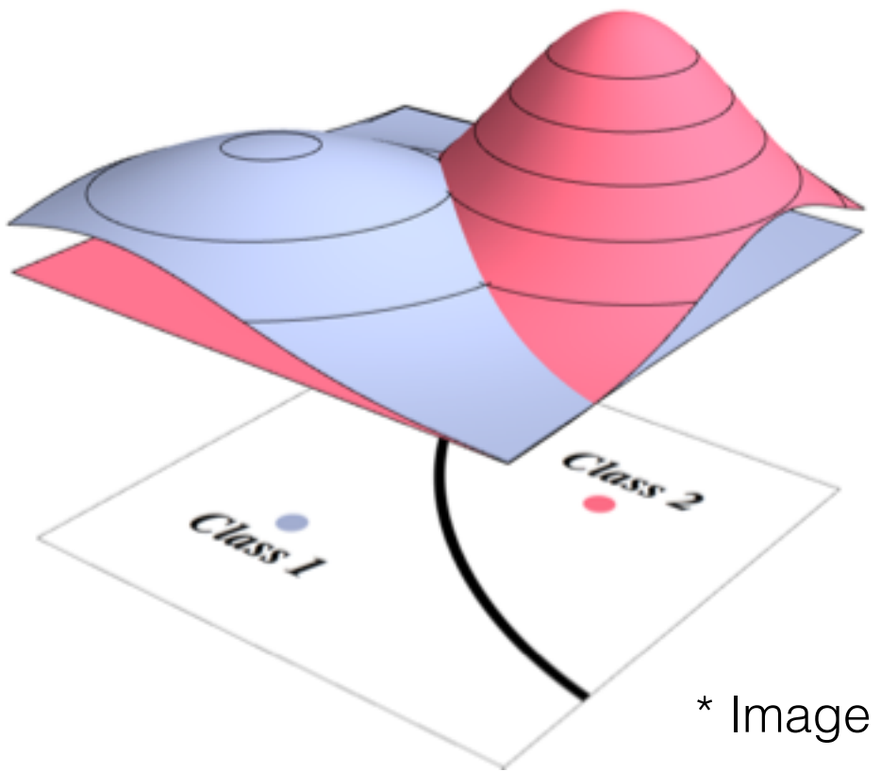


Generative models: an example with naive Bayes



* Image from Yaroslav Bulatov

Reminders/Comments

- Assignment 2 due this week
- Please see Touqir's clarifications
  - He is trying to make the outcomes more clear-cut
- Lectures are supposed to complement assignments
  - assignments are for you to learn, run into real problems first hand
- Final exam will only include topics in the notes
  - anything extra in lectures is for interest and context

Thought question

- Which optimization methods are more likely to overfit?
- Is it more likely for Stochastic GD or Batch GD to overfit?
- What about Second-order GD (i.e., Newton's method)?

Exercise questions

- What are the properties of different optimization approaches?
  - e.g., why are we taking the derivative and setting it to zero?
  - e.g., does the amount of data influence whether we obtain local or global solutions?
  - e.g., does stochastic gradient descent result in local solutions?
  - e.g., can gradient descent approaches only be used for convex problems?
  - e.g., why would you use second-order gradient descent rather than first-order gradient descent?

Batch GD versus SGD

- What are some advantages of batch gradient descent?
  - less noisy gradient
  - more clear strategies for selecting stepsize
- What are some advantages of SGD?
  - more computationally efficient: does not waste an entire epoch to provide an update to the weights
  - convenient for updating current solution with new data
- Does batch always give better solutions than SGD?

Classification so far

- Have been learning p(y | x)
  - Either for binary classification, as a Bernoulli
  - Or for multi-class classification, as a Multinomial
- These are called discriminative classifiers
  - i.e., learning a function of x, to predict distribution over y, i.e, f(x) = p(y | x)
  - Do not learn the distribution over x itself
  - **Note**: the classifiers have been linear so far, but this is not a requirement for discriminative classifiers, f can be a (highly) nonlinear function of x

Recall: Logistic regression

- Hyperplane $\mathbf{w}^\top \mathbf{x} = 0$ separates the two classes
  - P(y=1 | x, w) > 0.5 only when $\mathbf{w}^\top \mathbf{x} \geq 0$.
  - P(y=0 | x, w) > 0.5 only when P(y=1 | x, w) < 0.5, which happens when $\mathbf{w}^\top \mathbf{x} < 0$

What happens if w0 = 0?

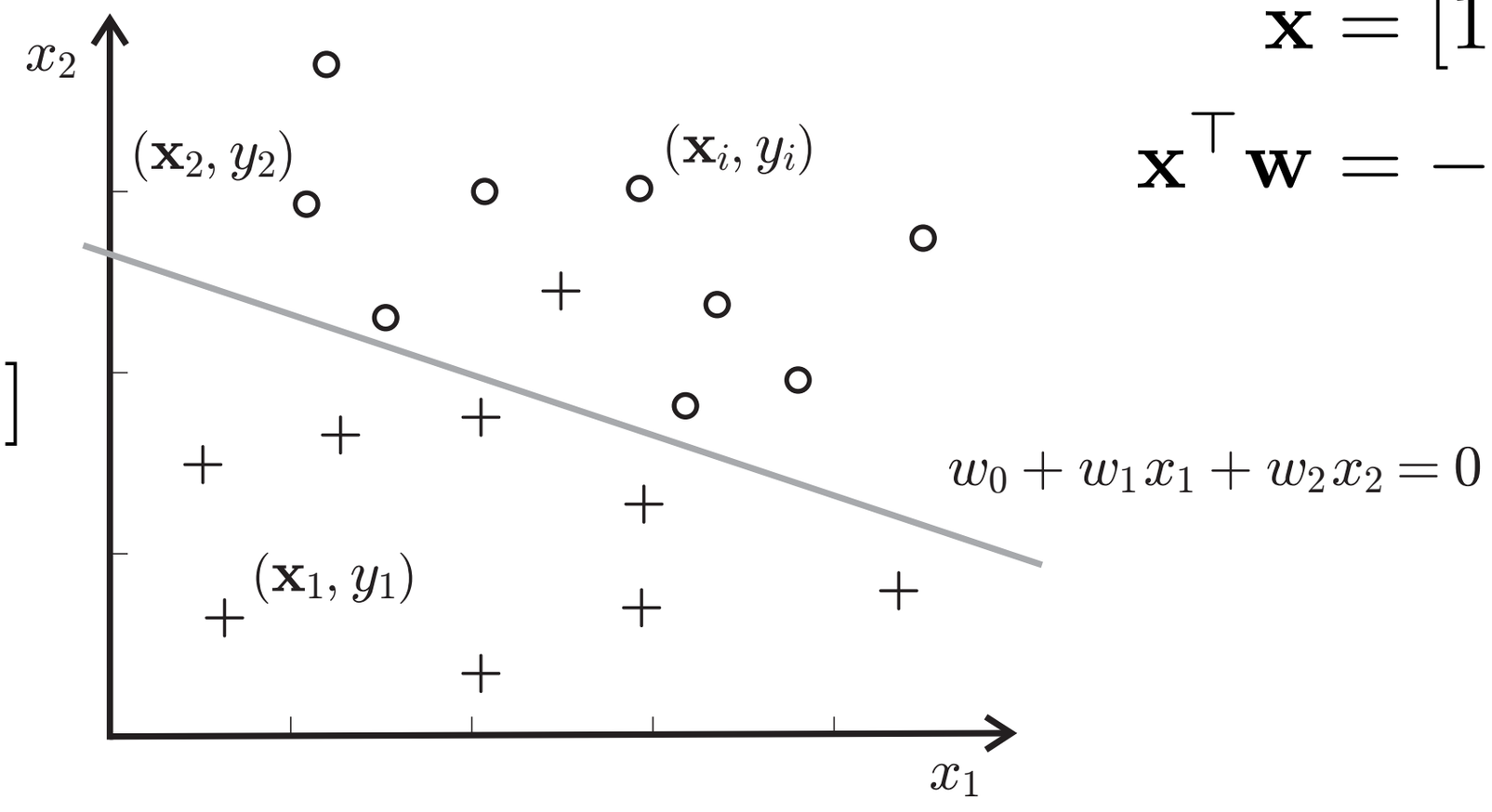
$\text{e.g., } \mathbf{w} = [2.75 \;\; -1/3 \;\; -1]$

$\mathbf{x} = [1 \;\; 4 \;\; 3]$

$\mathbf{x}^\top \mathbf{w} = -1.8$

$\mathbf{x} = [1 \;\; 2 \;\; 1]$

$\mathbf{x}^\top \mathbf{w} = 0.27$



Discriminative versus generative

- Discriminative: learn p(y | x), as a function of x

- In generative learning,
  - learn p(x | y) p(y) (which gives the joint p(x, y) = p(xly) p(y))
  - compute p(x | y) p(y), which is proportional to p(y | x)

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

- Question: how do we use p(xly) p(y) for prediction?

- Question: how do we learn p(xly) and p(y)?

- Question: why might we want to use generative models?

How to use generative models

- Our decision rule for using these probabilities is the same as with logistic regression: pick class with the highest probability

- Assume you have learned p(x l y) and p(y) from a dataset

- Compute

$$
\begin{aligned}
f(\mathbf{x}) &= \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \\
&= \arg\max_{y \in \mathcal{Y}} p(\mathbf{x}|y)p(y)/p(\mathbf{x}) \\
&= \arg\max_{y \in \mathcal{Y}} p(\mathbf{x}|y)p(y)
\end{aligned}
$$

How to learn generative models

- For discriminative, had to choose distribution over y given x
  - e.g. p(y | x) is Gaussian for continuous y
  - e.g. p(y | x) is Poisson for y in {1, 2, 3, …}
  - e.g. p(y | x) is Bernoulli for y in {0,1}
- Parameters to p(y | x) were w such that E[Y | x] = f(xw)
- Now we need to choose the distribution p(x | y) and p(y)
- How do we pick p(x | y) and p(y)? What are the parameters?

How do we pick distributions more generally?

- For p(x), picked Gaussian, Bernoulli, Poisson, Gamma
  - depending on the values x could take, or looking at a plot
- For conditional distributions, like p(y | x), we picked the distribution based on the properties of y
  - Once the given features x are fixed, are are just learning a distribution over y
- Imagine x is a 2-d Gaussian RV and y is a 1-d Gaussian
- What might you pick for p(x | y) and p(y)?

Another setting

- Imagine x is a 2-d Gaussian RV and y is a Bernoulli

- Now what might you pick for p(x | y) and p(y)?

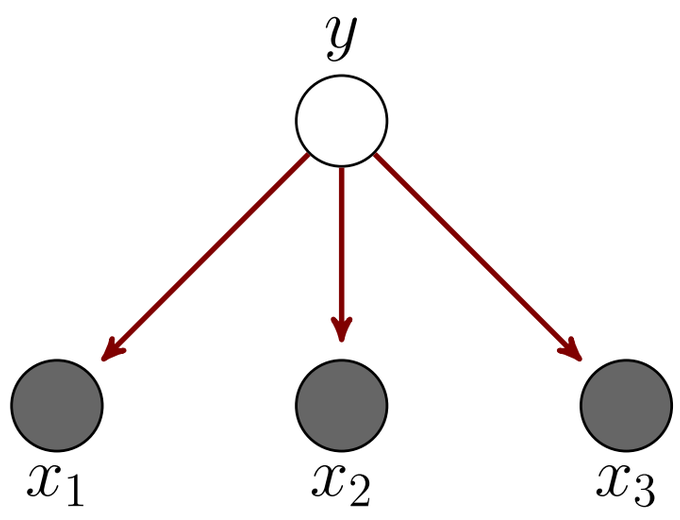
What if there are lots of features?

- Imagine x is a 1000-d random variable and y is a Bernoulli
- How can we determine what type of distribution to pick for x?
- What if we decide its a 1000-d multivariate Gaussian RV. Any issues with learning mean mu and covariance Sigma?

Simplifying assumptions

- How do we realistically learn p(x l y)?
- One option is to make a (strong) conditional independence assumption: the features are independent given the label

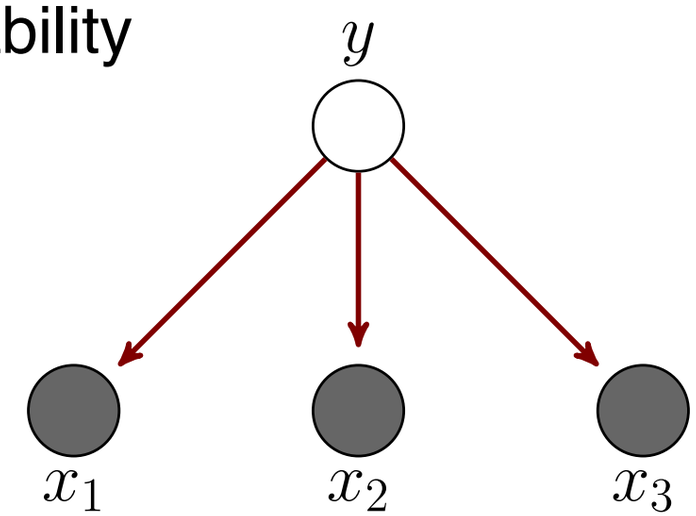
$$p(\mathbf{x}|y) = \prod_{i=1}^{d} p(x_i|y).$$



Conditional independence assumption

$$p(\mathbf{x}|y) = \prod_{i=1}^{d} p(x_i|y).$$

- Example: given a patient has the disease (y=1), attributes about patient are uncorrelated (e.g., age & smokes)
  - even within a class, age and smokes could be correlated
- Surprisingly, despite the fact that this seems unrealistic, in practice this can work okay
  - one hypothesis is we are running these algorithms on “easy” data
  - another is that dependencies skew the distribution equally across all classes, so no one class gets an increased probability



Naive Bayes

- For naive Bayes, we learn p(x | y) and p(y) exactly given this conditional independence assumption
  - For the classification setting

$$p(\mathbf{x}|y) = \prod_{i=1}^{d} p(x_i|y).$$

- Then we still need to choose p(xi | y) and p(y)
- What is p(y)? Table of values

$$p(y = 1), \ldots, p(y = k - 1), \text{ with } p(y = k) = 1 - \sum_{i=1}^{k-1} p(y = i)$$

	probability
1	p1
2	p2
.	.
.	.
.	.
k	pk

What if we have binary features?

- For binary features x in {0,1}, binary y in {0,1}, what is p(x l y)?

- How do we learn p(x l y)?

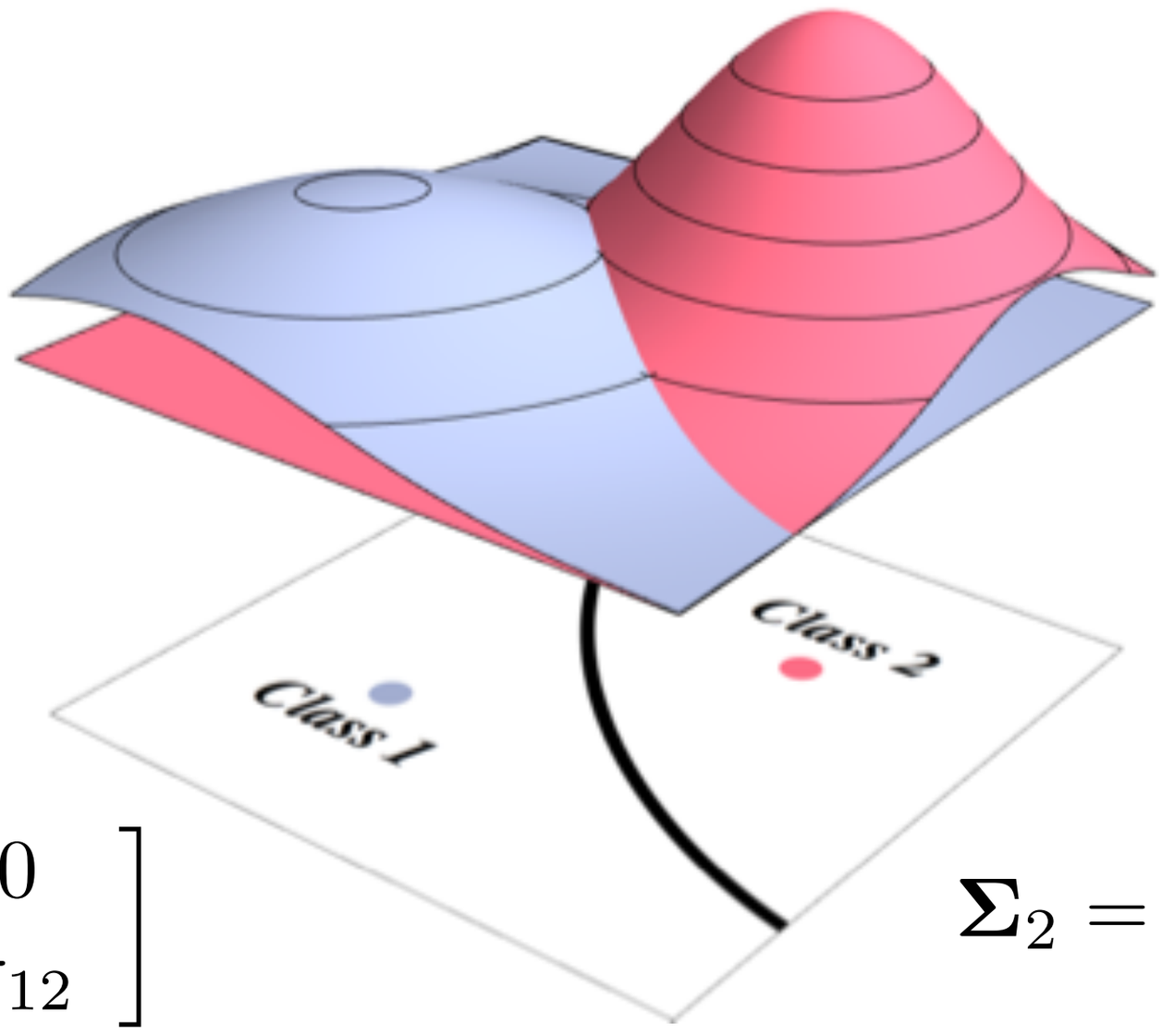
- How do we learn p(y)?

What if we have continuous features?

- For continuous features x, binary y in {0,1}, what could we choose for p(x | y)?
  - e.g., 5 features
- Need to identify p(xi | y) — what could this be?
- What are the parameters and how do we learn them?

Continuous naive Bayes

$\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1$



$\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} \sigma_{11} & 0 \\ 0 & \sigma_{12} \end{bmatrix}$$

$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} \sigma_{21} & 0 \\ 0 & \sigma_{22} \end{bmatrix}$$

* Image from Yaroslav Bulatov

Exercise

- Imagine someone gives you p(x l y) and p(y)
- They give you a test instance, with features x
  - e.g., x is an image, of pixel values (values between 0 to 255)
- Your goal is to predict the output y
  - e.g., if the image contains a cat or not
- How do you use p(x l y) and p(y) to make a prediction?

Whiteboard

- Naive Bayes derivation
- Exercise showing how to use naive Bayes
- Exercise dealing with missing values

Thought question

- “I understand that there is always a tradeoff between bias and variance of a model, and that this is a crucial part of model optimization, but does this mean that we can't ever achieve 100% accuracy, even in simple tasks like digit recognition?”
  - Recall reducible and irreducible error
  - Bias-variance is about selecting the model class (or function class)
  - Choosing the function class influences reducible error; more powerful function classes can reduce this error more
  - Irreducible error represents noise that we cannot model, and so we cannot ever achieve 100% accuracy
  - Why might such noise exist?

Exercise: Bias-variance for naive Bayes model

- Do you think naive Bayes has high bias and/or high variance?
  - is naive Bayes unbiased?
  - Recall: bias refers to error to the true function, in expectation
    - is f(y) = p(x l y) an unbiased estimate of f^*(y)?
- How do we compute bias and variance?
- naive Bayes can perform better in the small sample setting
  - see “On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes”, Ng and Jordan, 2002

Pros and Cons

- Discriminative
  - focus learning on the value we care about: p(y | x)
  - can be much simpler, particularly if features x complex: p(x | y) can be difficult to model without strong assumptions
- Generative
  - can be easier for expert to encode prior beliefs, e.g., for classifying trees in evergreen or deciduous, structure/distribution over the features (height, location) can be more clearly specified by p(x | y), whereas p(y | x) does not allow this information to be encoded
  - can sample from the generative model, obtain explanations
  - more amenable to missing values

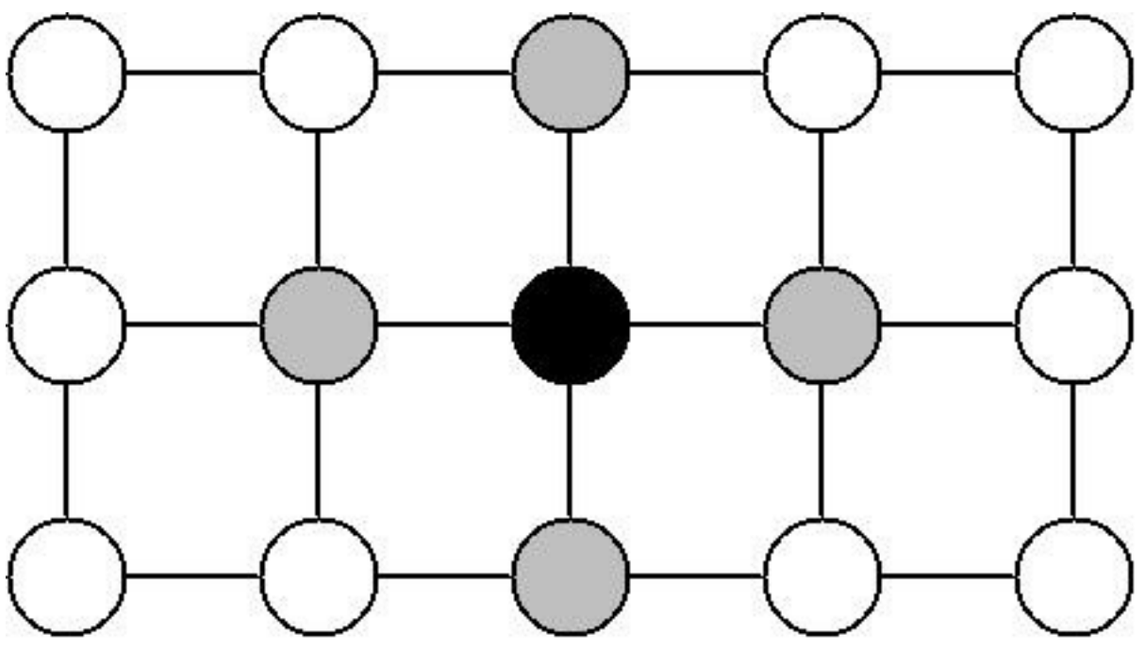
Sampling from generative model

- How sample (x,y) from generative model p(x,y) = p(x | y) p(y)
  - Sample y from p(y)
  - Then sample x from p(x | y)

- Why would you do this? Let's use the trees example again
  - Could sample trees from p(x | y) to see what your model produces
  - Could answer additional questions about the features, such as average leaf diameter in population
  - Could depict average tree, within a type or across types
  - Question: how would you do it within a type (deciduous or coniferous)?
  - Question: how would you do it across types?
  - More realistic example for explanations: obtain profile of typical person with heart disease

Other generative models

- Often use generative models for images (markov random field)
  - e.g., can model the spatial structure in the distribution p(x l y)
  - We know a lot about image structure, can take advantage of that expert knowledge in choice of p(x l y)



Thought question

- Is it possible to provide a prediction and an estimate of how confident we are in that prediction?
  - “Let's say the data you are modeling is inherently high-variance and it may not be appropriate to give exact estimates for new data. Is it possible to generate a distribution range, i.e., the outcome will likely fall within some N(mu, sigma^2)”
- For p(y | x) Gaussian, we could try to estimate mean mu = <x,w> and estimate variance sigma^2
  - when predict E[Y | x], have a sense of how much y can vary
- But what if our estimates are wrong? To be confident in predictions, want estimate of confidence in parameters

Practical considerations: incorporating new data

- Imagine you have a discriminative model (say weights w) and now want to incorporate new data
- How can you do this with regression approaches?
- Option 1: add data to your batch and recompute entire solution
- Option 2: start from previous w (warm start), add data to your batch and do a few gradient descent steps
- Option 3: use stochastic gradient descent update and step in the direction of the gradient for those samples
  - for a constant stream of data, will only ever use one sample and then throw it away

How about naive Bayes?

- How can we update means and covariances in naive Bayes model?

- Learned mu_{j,c} and sigma_{j,c} for each feature j, each class c

Updating naive Bayes model

- Keep a running average of mu_{j,c} and sigma_{j,c}, with number of samples n_{j,c} for feature j class c
- For a new sample (x, y), update parameters mu_{j,y} and sigma_{j,y} using:
  - mu_{j,y} = mu_{j,y} + (x_j - mu_{j,y})/ n_{j,c}
  - sndmoment_{j,y} = sndmoment_{j,y} + (x_j^2 - sndmoment_{j,y})/ n_{j,c}
  - sigma_{j,c} = sqrt(sndmoment_{j,y} - mu_{j,y}^2)

Exercise: Non-stationarity

- A theme in thought questions: “What if the distribution changes over time?”
- Say you have trained your model so far, and now are getting new data that is from a slightly different distribution (drift)
  - Data could be coming from a physical system that wears out, such as a robot vacuum collecting data where its wheels wear-out
  - Data could reflect continually (but slowly) changes preferences in human population
- How can we handle this?
  - We can update parameters with this new data, and hope that works
  - How can we give more weight

Exercise: Non-stationarity

- A theme in thought questions: “What if the distribution changes over time?”
- How can we handle this?
  - Don’t want to throw away previous solution, since drift is slow so old solution is still (mostly) reflective
  - We can update parameters with this new data, and hope that works
  - How can we give more weight to new data? Say for naive Bayes

Exponential average

$$\begin{aligned}\mu_t &= \alpha x_t + (1-\alpha)\mu_{t-1} \\ &= \alpha x_t + (1-\alpha)(\alpha x_{t-1} + (1-\alpha)\mu_{t-2}) \\ &= \alpha x_t + \alpha(1-\alpha)x_{t-1} + (1-\alpha)^2(\alpha x_{t-2} + (1-\alpha)\mu_{t-3}) \\ &= \alpha \sum_{i=0}^{\infty}(1-\alpha)^i x_{t-i}\end{aligned}$$

where $0 < \alpha < 1$

$$\sum_{i=0}^{\infty}(1-\alpha)^i = \frac{1}{\alpha}$$