COMPUTER SCIENCE
INDIANA UNIVERSITY
School of Informatics and Computing
Bloomington

Hidden variable models

Reminders/Comments

- Not requiring the assignment to be written-up in tex, but if writing it by hand, it needs to be legible
- Small comment: can equivalently use
  - diagonal matrix and standard matrix multiplication: C v
  - element-wise multiplication (Hadamard product) of vectors: c circa v
- Naive Bayes question asks about adding a column of ones; I may have provided a chunk of code that is robust to this, but answer the question assuming I had not done so
- Questions about accuracy of the algorithms in the assignment

Accuracy of learned models

- The datasets from UCI are somewhat notoriously simple
  - "Very Simple Classification Rules Perform Well on Most Commonly Used Datasets", Holte, 1993
- They have gotten more interesting, but still some issues
- For many machine learning algorithms, the differences are only evident on some datasets (not any dataset)
- Here the goal is to implement the algorithms and try to ensure their correctness
- Question 3: adding regularizers *can* outperform the base logistic regression; if it is not, try to see why and explain
  - look at the (final) weights as a debugging tool
  - print out the function values that are obtained along the way, ensure they steadily improving

Question 3 and regularizers

- Adding regularizers *can* outperform the base logistic regression

- If it is not, try to see why and explain
  - look at the (final) weights as a debugging tool
  - print out the function values that are obtained along the way, ensure they steadily improving
  - why should they be steadily improving?
  - think about your range of regularizers and what it *should* be; for example, how did your choice of l2 regularizer affect the solution in linear regression?

Neural networks

- Using neural networks *can* (and will if tuned well) outperform the base logistic regression
- If it is not, try to understand why
  - again, look at the (final) weights as a debugging tool
  - in this case, should you objective value be steadily decreasing? Is this true for batch gradient descent or stochastic gradient descent?
- For all your algorithms, consider comparing to python's library as a sanity check
  - if their learned models are significantly outperforming your learned models, then you might have a bug
  - if their models perform similarly poorly, then this might simply be a hard problem for that algorithm and/or tuning is difficult
  - if your model out-does python, feel proud and don't be too surprised; a capable implementer can often outperform packages

Student example for neural net

- "Strange" behavior in neural network

- I ran it with stepsize = .001 with the following iterations and accuracies:
  - 2: 50%
  - 3: 63%
  - 4: 68%
  - 5: 70% <--peak
  - 6: 60%
  - 7: 40%
  - 8: 48%
  - 9: 67% <--another peak
  - 10: 47%
  - 11+ ~50%

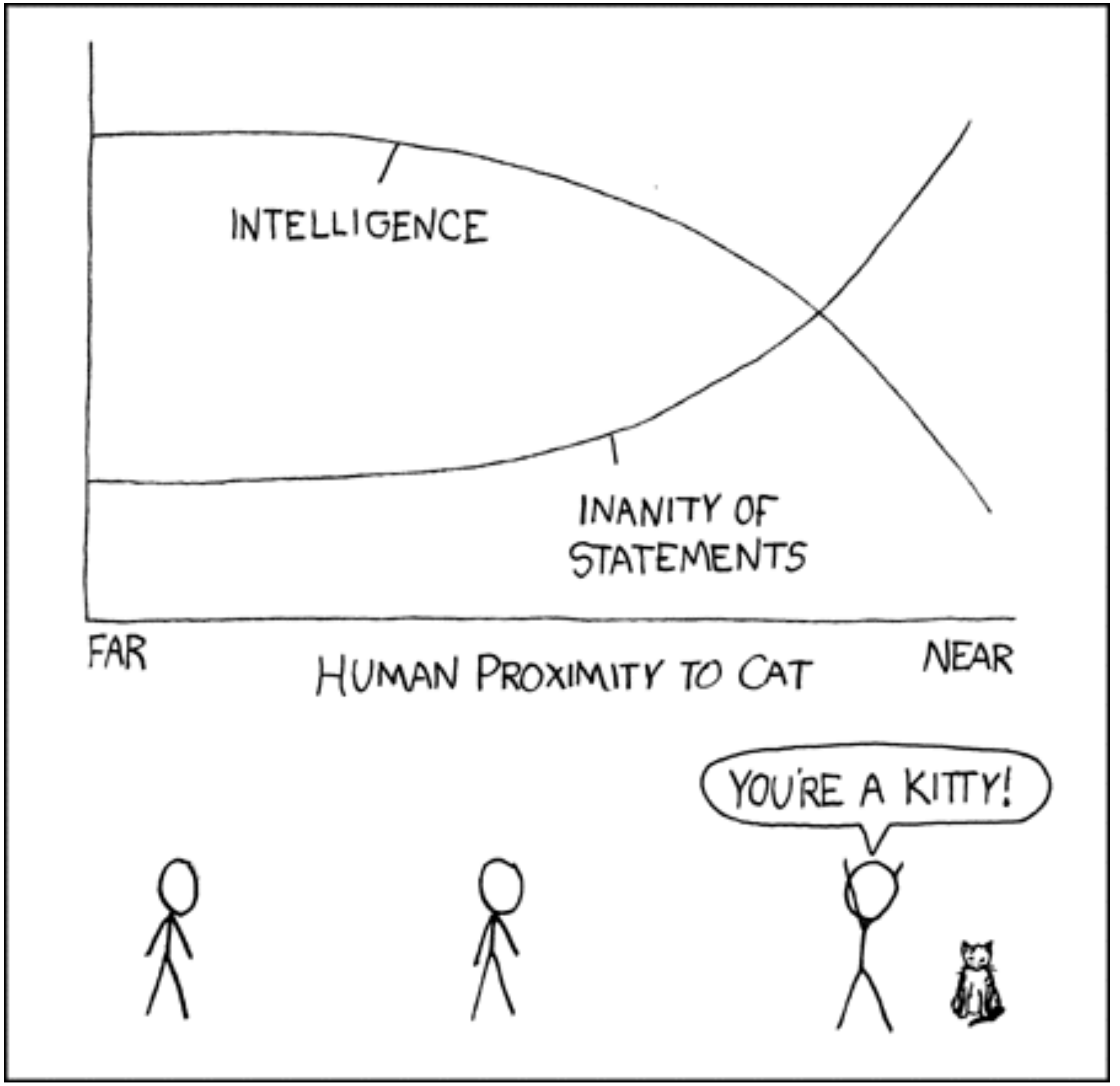
- I then ran it 5 times with stepsize = .00005 with 100 iterations and got the following accuracies: 80%, 45%, 63%, 73%, 73%.

Hidden variables

- Different from missing variables, in the sense that we *could* have observed the missing information
  - e.g., if the person had just filled in the box on the form
- Hidden variables are never observed; rather they are useful for model description
  - e.g., hidden, latent representation
  - e.g., hidden state that drives dynamics
- Hidden variables make specification of distribution simpler
  - p(x | D) = \int_h p(x | D, h) p(h)
  - p(x | D, h) is often much simpler to specify

Intuitive example

- Underlying “state” influencing what we observe; partial observability makes what we observe difficult to interpret
- Image we can never see that a kitten is present; but it clearly helps to explain the data



Hidden variable models

- Probabilistic PCA and factor analysis
    - common in psychology
- Mixture models
- Hidden Markov Models
    - commonly used for NLP and modeling dynamical systems

Probabilistic PCA

- In PCA, we learned p(x I D, h)
  - What were the assumptions on p(x I D, h)?
- For Probabilistic PCA, we learn p(x I D)
- Given some prior p(h), we have

$$p(\mathbf{x}|\mathbf{D}) = \int_{\mathcal{H}} p(\mathbf{x}|\mathbf{D}, \mathbf{h}) p(\mathbf{h}) d\mathbf{h}$$

Modified goal

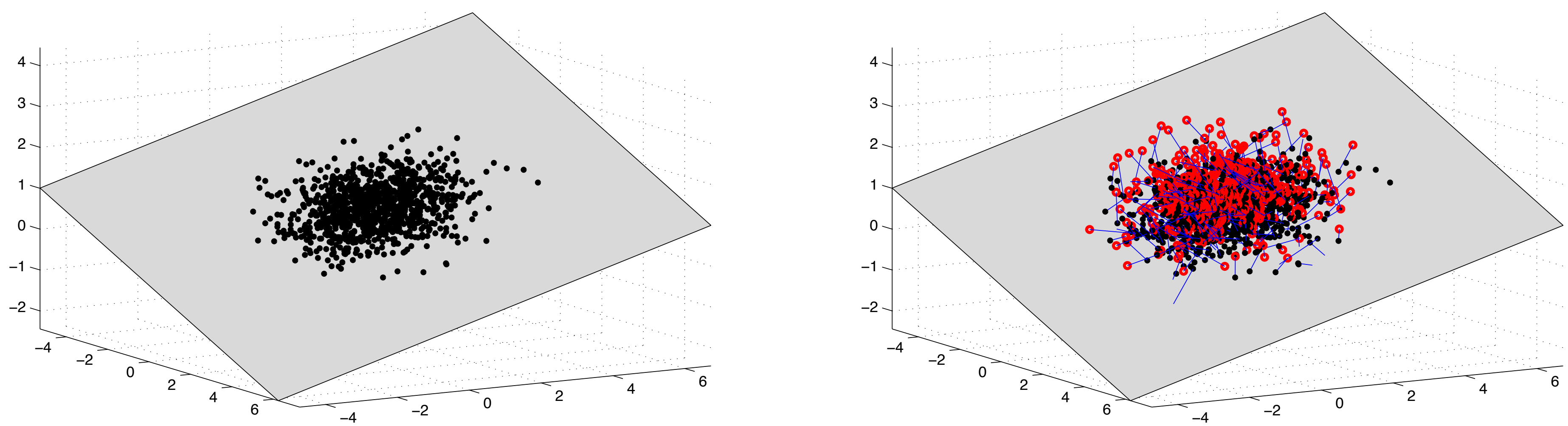
- The interpretation of the hidden factors as a new representation is still reasonable in this setting
- Now our goal is to obtain a distribution over x, only given the dictionary and not the representation
  - Why do we care about having distributions over x? Why isn't p(x I D, h) "good" enough?
  - What can we do with p(x I D) that we could not do with p(x I D, h), assuming we have learned D?

Resulting differences

- Resulting solution for D is actually very similar, in the case of probabilistic PCA
  - In PCA, D = U Sigma
  - In probabilistic PCA, D = U (Sigma - sigma^2 I)
- The model now is generative
  - can think of the previous one as discriminative, since need h to obtain the distribution over x
  - parallels p(y I x) versus p(x,y)
- Solution approach is different
- Probabilistic PCA extends to more generally to other probabilistic models (e.g. factor analysis) that does not have such a similar solution

Generating data

- Sample h from p(h), then sample x from p(x I D, h)
  - both of these distributions are Gaussian and so simple to sample



(a) (b)

Figure : Factor Analysis: 1000 points generated from the model. **(a)**: 1000 latent two-dimensional points $\mathbf{h}^n$ sampled from $\mathcal{N}(\mathbf{h}|\mathbf{0},\mathbf{I})$. These are transformed to a point on the three-dimensional plane by $\mathbf{x}_0^n = \mathbf{c} + \mathbf{F}\mathbf{h}^n$. The covariance of $\mathbf{x}_0$ is degenerate, with covariance matrix $\mathbf{F}\mathbf{F}^{\mathsf{T}}$. **(b)**: For each point $\mathbf{x}_0^n$ on the plane a random noise vector is drawn from $\mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\boldsymbol{\Psi})$ and added to the in-plane vector to form a sample $\mathbf{x}^n$, plotted in red. The distribution of points forms a 'pancake' in space. Points 'underneath' the plane are not shown.

Other hidden variable models

- Probabilistic PCA and factor analysis
  - common in psychology
- Mixture models
- Hidden Markov Models
  - commonly used for NLP and modeling dynamical systems

Gaussian mixture model

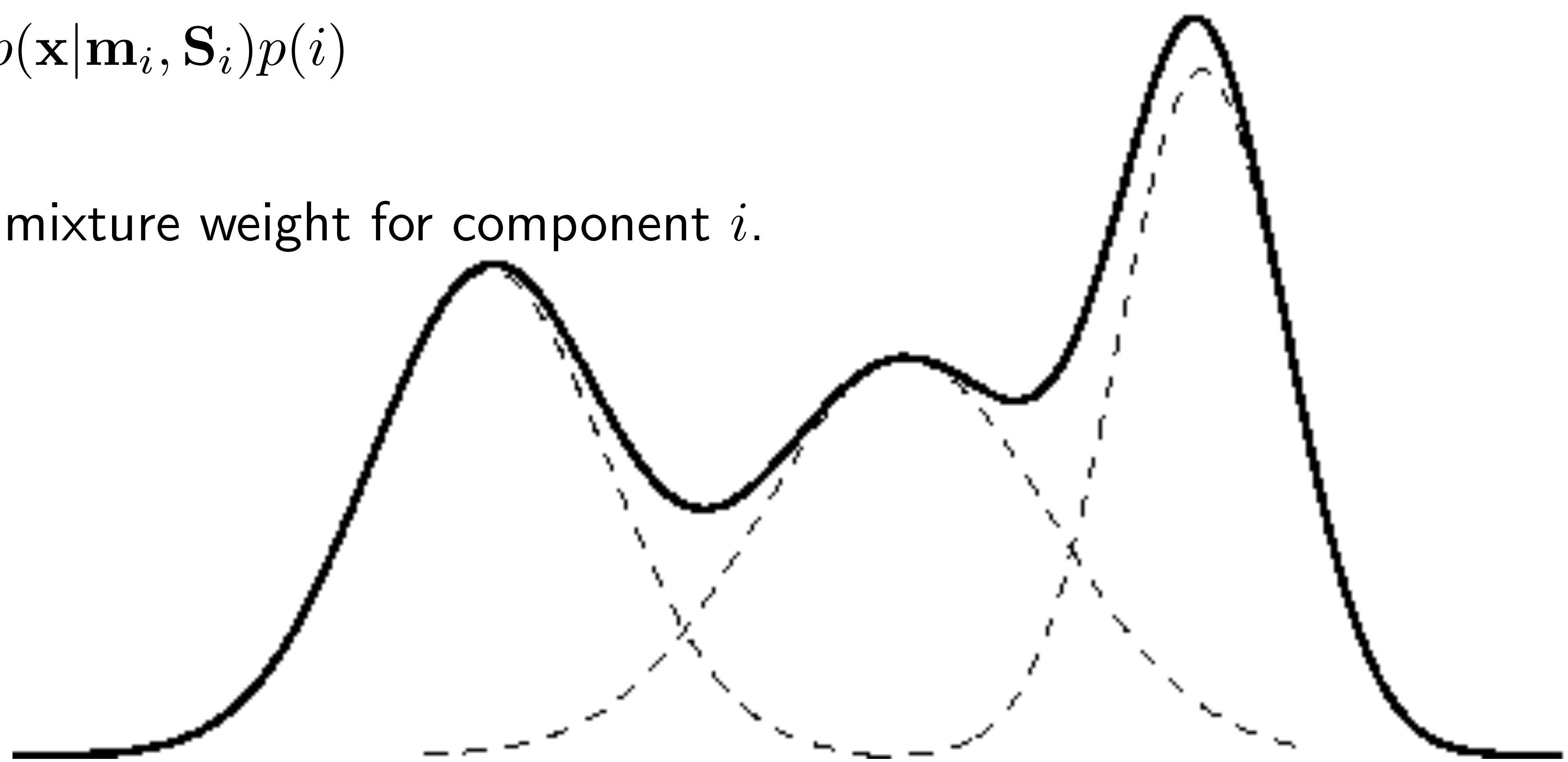
A D dimensional Gaussian distribution for a continuous variable $\mathbf{x}$ is

$$p(\mathbf{x}|\mathbf{m},\mathbf{S}) = \frac{1}{\sqrt{\det(2\pi\mathbf{S})}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\mathsf{T}}\mathbf{S}^{-1}(\mathbf{x}-\mathbf{m})\right\}$$

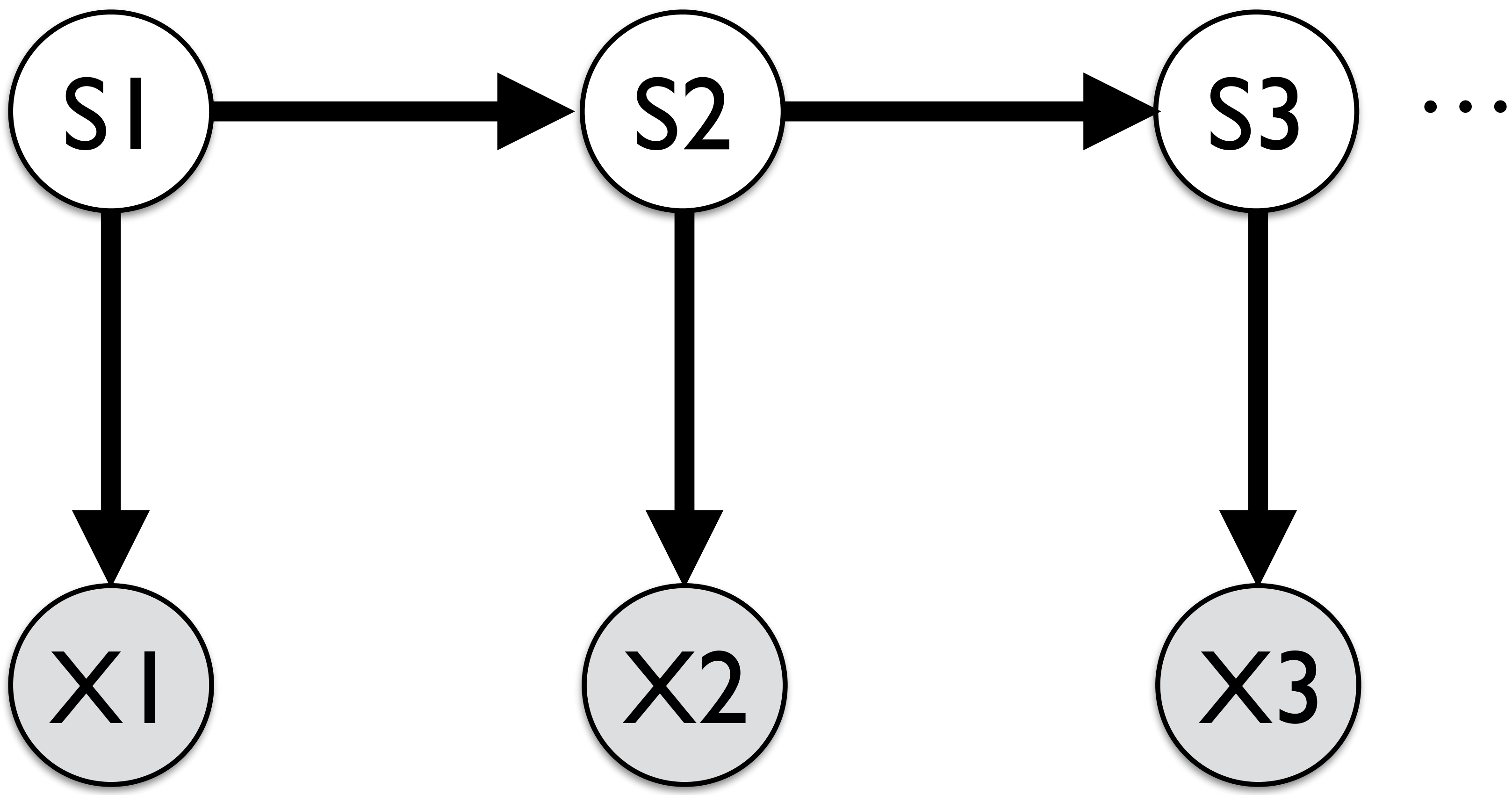
where $\mathbf{m}$ is the mean and $\mathbf{S}$ is the covariance matrix. A mixture of Gaussians is then

$$p(\mathbf{x}) = \sum_{i=1}^{H} p(\mathbf{x}|\mathbf{m}_i,\mathbf{S}_i)p(i)$$

where $p(i)$ is the mixture weight for component i.



Hidden Markov Model



The observation are x1, x2, x3
Temporally related
Dynamics driven by hidden state

Closed-form solutions

- For some hidden variable models, have a closed form solution
  - probabilistic PCA
  - factor analysis

- Probabilistic PCA solution:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

$$\mathbf{D} = \mathbf{U}_k(\mathbf{\Sigma}_k^2 - \sigma^2\mathbf{I}_k)^{1/2}$$

$$\sigma^2 = \frac{1}{d-k}\sum_{i=k+1}^{d}\sigma_i^2$$

Expectation-maximization

- We can use an expectation-maximization approach instead to incrementally compute the solution (rather than a closed form)
- Similar to alternating descent approach taken for RFMs
  - For PCA, instead of computing a closed-form solution to D and H, we could have simply used gradient descent with our objective
- What is the advantage to using the incremental EM approach, when we already have a closed form?
  - other than as an educational example of EM

Whiteboard

- Closed form solution for probabilistic PCA
- Expectation-maximization for probabilistic PCA