Announcements

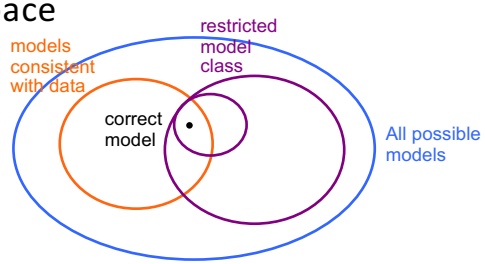
- No homework 3, Kaggle and Lab 4 were due on Sunday
- Tests are almost graded
  - There will be a big curve so don't worry too much about your grades
- There will be an option in a *few weeks* for a take home test for a bit of extra credit.
  - If you are above the median and have turned in labs and homeworks please don't do this—you don't need the credit.

#	Δrank	Team Name	Score	Entries	Last Submission UTC (Best – Last Submission)
1	–	KU_EECS_800_Kaggle1	872.76513	57	Mon, 27 Mar 2017 05:57:58
2	↑15	Tejaswini	873.42802	25	Mon, 27 Mar 2017 00:53:49
3	↑7	Niharika	888.78806	18	Mon, 27 Mar 2017 04:17:19 (-10.1h)
4	↓2	d923m604	895.61996	17	Mon, 27 Mar 2017 00:55:44
5	↑6	s148p508	897.03955	11	Mon, 27 Mar 2017 04:00:42
6	↑3	Madhu Chegondi	922.30486	13	Mon, 27 Mar 2017 03:53:38 (-0.1h)
7	↓1	Rahul Kakani	939.90934	27	Sun, 26 Mar 2017 21:12:50
8	↑23	dmenager	992.50564	7	Sun, 26 Mar 2017 20:48:22 (-38.7h)
9	↑20	Jbondoc	1002.31348	15	Mon, 27 Mar 2017 00:14:17 (-28.7h)
10	↑23	james_rolfe	1004.75114	15	Fri, 24 Mar 2017 20:38:28 (-0.3h)
11	↑19	Venkat Vaddula	1007.47823	24	Mon, 27 Mar 2017 03:38:36 (-2.4d)
12	↑26	taylorg	1008.33863	6	Thu, 23 Mar 2017 00:16:37 (-0.8h)
13	↑23	Lokesh	1009.96434	10	Mon, 27 Mar 2017 00:51:26 (-23.8h)
14	↑9	KiaKiaee	1011.82388	49	Mon, 27 Mar 2017 04:12:13 (-27.9h)
15	↑12	PaulM	1012.59730	8	Mon, 27 Mar 2017 03:21:25 (-0.3h)
16	↑21	Eric Smith	1012.67576	28	Mon, 27 Mar 2017 00:33:48 (-21h)

Regularization in Neural Networks

Continuation from last lecture

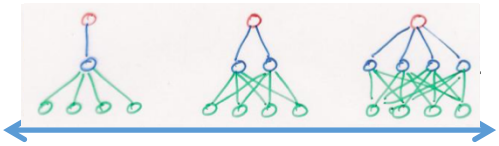
Model Space



- Challenge for learning
  - Start with model class appropriately restricted for problem domain

Model Complexity

- Models range in their flexibility to fit arbitrary data



simple model — high bias — low bias — complex model

constrained — low variance — high variance — unconstrained

small capacity may prevent it from representing all structure in data

large capacity may allow it to memorize data and fail to capture regularities

Overfitting

- Neural networks can easily overfit data
  - There's a lot of parameters (each weight)
- How can we avoid overfitting?
  - Weight- decay
  - Weight regularization
  - Early stopping
  - Model averaging
  - Dropout

Regularization Techniques

- Instead of starting with smallest net possible, use a larger network and apply various tricks to avoid using the full network capacity
- 7 ideas to follow…

Regularization Techniques

1. early stopping
   - Rather than training network until error converges, stop training early
     - Rumelhart
       - hidden units all go after the same source of error initially -> redundancy
     - Hinton
       - weights start small and grow over training
       - when weights are small, model is mostly operating in linear regime
   - Dangerous: Very dependent on training algorithm
     - e.g., what would happen with random weight search?
   - While probably not the best technique for controlling model complexity, it does suggest that you shouldn't obsess over finding a minimum error solution.

When To Stop Training ('regularization' 1)

- A. Train *n* epochs; lower learning rate; train *m* epochs
  - bad idea: can't assume one-size-fits-all approach
- B. Error-change criterion
  - stop when error isn't dropping
  - My recommendation: criterion based on % drop over a window of, say, 10 epochs
    - 1 epoch is too noisy
    - absolute error criterion is too problem dependent

When To Stop Training ('regularization' 1)

- C. Weight-change criterion
  - Compare weights at epochs t-10 and t and test:

$$\max_i \left| w_i^t - w_i^{t-10} \right| < \theta$$

  - Don't base on length of overall weight change vector
  - Possibly express as a percentage of the weight
  - Be cautious: small weight changes at critical points can result in rapid drop in error

Regularization Techniques

2. Weight penalty terms

L2 weight decay

$$E = \frac{1}{2}\sum_j \left(t_j - y_j\right)^2 + \frac{\lambda}{2}\sum_{i,j} w_{ji}^2$$

$$\Delta w_{ji} = \varepsilon \delta_j x_i - \varepsilon \lambda w_{ji}$$

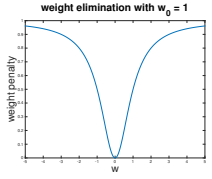
L1 weight decay

$$E = \frac{1}{2}\sum_j \left(t_j - y_j\right)^2 + \frac{\lambda}{2}\sum_{i,j} \left| w_{ji} \right|$$

$$\Delta w_{ji} = \varepsilon \delta_j x_i - \varepsilon \lambda \text{sign}(w_{ji})$$

weight elimination

$$E = \frac{1}{2}\sum_j \left(t_j - y_j\right)^2 + \frac{\lambda}{2}\sum_{i,j} \frac{w_{ji}^2 / w_0^2}{1 + w_{ji}^2 / w_0^2}$$



Regularization Techniques

3. Hard constraint on weights
   - Ensure that $\sum w_{ji}^2 < \phi$ for every unit
   - If constraint is violated, rescale all weights: $w_{ji} \leftarrow w_{ji} \frac{\phi}{\sum_i w_{ji}^2}$
   - I'm not clear why L_2 normalization and not L_1

4. Injecting noise
   - Not covered here…

Regularization Techniques

6. Model averaging
   - Ensemble methods
   - Bayesian methods

7. Drop out

More On Dropout

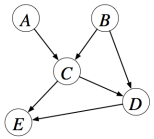
- With H hidden units, each of which can be dropped, we have 2^H possible models
- Each of the 2^{H-1} models that include hidden unit h must share the same weights for the units
  - serves as a form of regularization
  - makes the models cooperate
- Including all hidden units at test with a scaling of 0.5 is equivalent to computing the geometric mean of all 2^H models
  - exact equivalence with one hidden layer
  - "pretty good approximation" according to Geoff with multiple hidden layers

Graphical representations of models

A quick aside to explain figures in book and various research papers

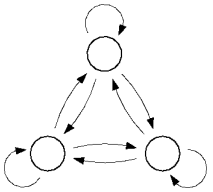
Graphical Models

- Nodes and edges
  - Nodes represent random variables
  - Edges represent statistical dependence between random variables
    - Conditional dependence



Graphical Models

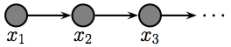
- Advantages:
  - Structure brings statistical and computational efficiencies
  - Less data needed
  - Less time to perform inference



Examples: Markov chains

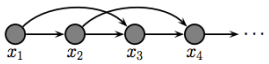
- First order Markov chain

$$p(x_{1:T}) = p(x_1) \prod_{t=2}^{T} p(x_t|x_{t-1})$$



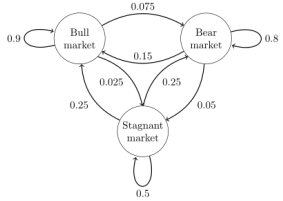
- Second order Markov chain

$$p(x_{1:T}) = p(x_1, x_2) \prod_{t=3}^{T} p(x_t|x_{t-1}, x_{t-2})$$



Examples: Markov chains

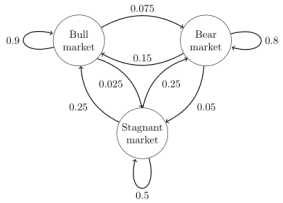
- Given that today was a bull market, what is the probability that tomorrow will be a bear market?
- Can look forward and backward in time.
- Further from current state, more uncertainty



Examples: Markov chains

- Given that today was a bull market, what is the probability that tomorrow will be a bear market?
- Can look forward and backward in time.
- Further from current state, more uncertainty

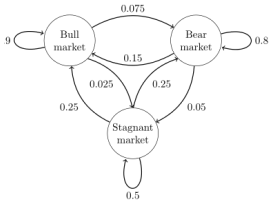
$$P = \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}$$



Examples: Markov chains

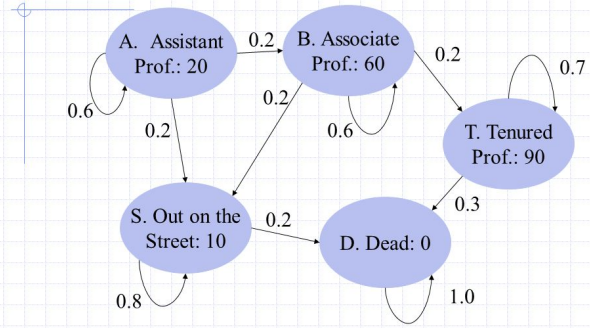
- Given that today was a bull market, what is the probability that tomorrow will be a bear market?
- Can look forward and backward in time.
- Further from current state, more uncertainty

$$\begin{aligned} x^{(n+3)} &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}^3 \\ &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.7745 & 0.17875 & 0.04675 \\ 0.3575 & 0.56825 & 0.07425 \\ 0.4675 & 0.37125 & 0.16125 \end{bmatrix} \\ &= \begin{bmatrix} 0.3575 & 0.56825 & 0.07425 \end{bmatrix}. \end{aligned}$$



Courtesy of Michael Littman

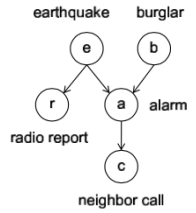
Example: Academic Life



What is the expected lifetime income of an academic?

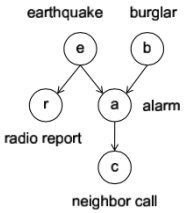
Examples: Bayes Nets

- Lets say an alarm will go off only if there is a burglar or an earthquake.
- Additionally, the alarm will sometimes result in a call from the neighbor.
- The earthquake will also generate a radio report.
- What do we know about the world if we hear an alarm?



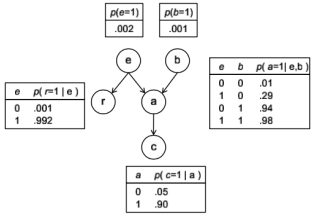
Examples: Bayes Nets

- If the alarm is going off what's the probability of an earthquake?
- We can think of these arrows as causal.
- Allows for probabilistic reasoning about causes and effects.



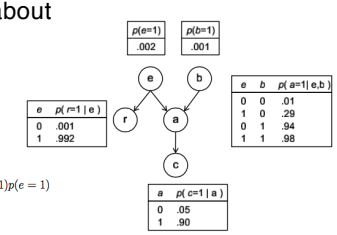
Examples: Bayes Nets

- If the alarm is going off what's the probability of an earthquake?
- We can think of these arrows as causal.
- Allows for probabilistic reasoning about causes and effects.



Examples: Bayes Nets

- If the alarm is going off what's the probability of an earthquake?
- We can think of these arrows as causal.
- Allows for probabilistic reasoning about causes and effects.



$$p(a=1|b=1) = p(a=1|b=1,e=0)p(e=0) + p(a=1|b=1,e=1)p(e=1)$$
$$= (0.94)(0.998) + (0.98)(0.002)$$
$$= 0.9401$$

What does this buy us?

- All the advantages of Bayesian models.
  - Prior knowledge is embedded in the graphical structure
  - Little data can still allow for meaningful predictions
  - Inference techniques
- Limited parameters but still lots of power
- We can abstract further, getting rid of even more parameters
  - We'll see this by introducing latent variables

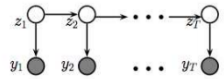
Why do we need this?

- For one, lots of algorithms are designed from this type of thinking.
  - Decision trees
  - Naïve Bayes classifiers
  - Hidden Markov Models
- Unsupervised learning techniques

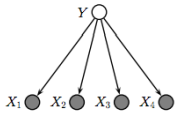
- Inference over latent (unobserved) states.
- Exploration of causality.
- Formalization of a class of models.

Some standard algorithms

- Hidden Markov models



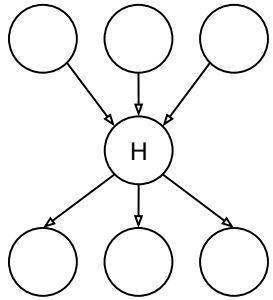
- Naïve Bayes Classifier



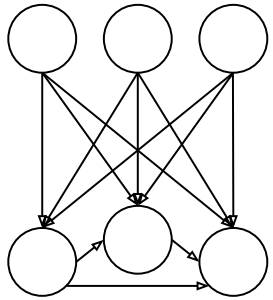
Gaussian Mixture Models

Latent Variable Models

- We often see dependence between variables in our data
- We can assume that the dependence can be captured by imposing structure in our model (e.g. NB, LDA etc.)
- Many of the basic (graphical) models assume that the *observed* variables are correlated with each other.
- What if instead we assume that there's a "hidden" common cause?
- Then we're working with latent (hidden) variable models



17 parameters



59 parameters

Latent Variable Models (LVMs)

- Just like most topics in this class, there are entire courses on this
- Here we introduce just one class of LVMs

- Mixture models
  - Interesting case to highlight a very powerful model **GMM**
  - Also useful to illustrate an important algorithm **EM**

Mixture models

- A simple form of LVM

- Assume there is a latent state indicating a particular class
- Then the probability of a given observation is just a combination of how likely a particular class is and then how likely that data would be under that class.
  - E.g. We have computer science students and English students. We observe their scores on math exams. A particular set of scores is a weighted sum of how likely a particular score would be for a typical engineering student and (separately) for a particular english student

Mixture models

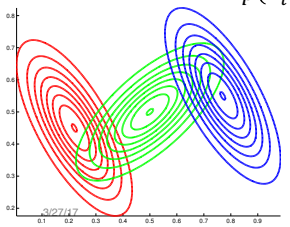
- A simple form of LVM
- Assume there is a latent state indicating a particular class
- Then the probability of a given observation is just a combination of how likely a particular class is and then how likely that data would be under that class.

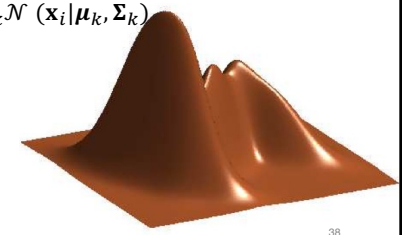
$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}_i|\boldsymbol{\theta})$$

Mixture of Gaussians

- Assume that each of the *k* classes is distributed as a MVN
- Our model is then redefined as

$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$





The power of Gaussians

- We know that a Gaussian distribution is a maximum entropy distribution
  - This means it's very expressive
- In fact, given a sufficiently large number of mixture components, a GMM can be used to approximate any density defined on $\mathbb{R}^p$

Using mixture models

- First use case: Use them to model $p(\mathbf{x_i})$ directly
  - This is **unsupervised** and can be used for
    - Data compression
    - Outlier detection
    - Generative classifiers
  - Book says this isn't common but I think it is very widely used nowadays
- Second use case: clustering
  - **Supervised**
    - Fit a model, compute the probability of a class for new data given the learned parameters
    - Can compute the "responsibility" (think contribution) of each cluster on a particular observation

Using mixture models

- Model $p(\mathbf{x_i})$ directly
  - We don't know the true class assignments
- Clustering
  - We see the class labels which we assume to be the latent class assignment

$$p(z_i = k|\mathbf{x}_i, \boldsymbol{\theta}) = \frac{p(z_i = k|\boldsymbol{\theta})\, p(\mathbf{x}_i|z_i = k, \boldsymbol{\theta})}{\sum_{k'=1}^{K} p(z_i = k'|\boldsymbol{\theta})\, p(\mathbf{x}_i|z_i = k', \boldsymbol{\theta})}$$

- The only difference in these two cases is during training
  - Whether we know the labels or not
  - Or if we have to iteratively guess the "true" labels and update the parameters