Connectionist Computing
COMP 30230/41390

Gianluca Pollastri

office: E0.95, Science East.

email: gianluca.pollastri@ucd.ie

Credits

- **Geoffrey Hinton, University of Toronto.**
  - **borrowed some of his slides for "Neural Networks" and "Computation in Neural Networks" courses.**

- **Ronan Reilly, NUI Maynooth.**
  - **slides from his CS4018.**

- **Paolo Frasconi, University of Florence.**
  - **slides from tutorial on Machine Learning for structured domains.**

Lecture notes on Brightspace

- **Strictly confidential...**
- **Slim PDF version will be uploaded later, typically the same day as the lecture.**
- **If there is demand, I can upload onto Brightspace last year's narrated slides.. (should be very similar to this year's material)**

Books

- No book covers large fractions of this course.

- Parts of chapters 4, 6, (7), 13 of Tom Mitchell's "Machine Learning"
- Parts of chapter V of Mackay's "Information Theory, Inference, and Learning Algorithms", available online at:

http://www.inference.phy.cam.ac.uk/mackay/itprnn/book.html

- Chapter 20 of Russell and Norvig's "Artificial Intelligence: A Modern Approach", also available at:

http://aima.cs.berkeley.edu/newchap20.pdf

- More materials later..

Marking

- **3 landmark papers to read, and submit a 10-line summary on Brightspace about: each worth 6-7%**
- **a connectionist model to build and play with on some sets, write a report: 30%**
- **Final Exam in the RDS (50%)**

Programming assignment

- Implement a Multi-Layer Perceptron, test it.
- The description on Brightspace.
- Submit through Brightspace code and test results by Dec the 5th at 23:59, any time zone of your choice (Baker Island?).
- 30% of the overall mark
- One third of a grade down every day late, that is: if you deserve an A and you're 1 day late you get an A-, 2 days late a B+, etc.

Back to learning

- **Supervised Learning (this models p(output|input))**
  - **Learn to predict a real valued output or a class label from an input.**
- **Unsupervised Learning (this models p(data))**
  - **Build a causal generative model that explains why some data vectors occur and not others**

    or
  - **Learn an energy function that gives low energy to data and high energy to non-data**

    or
  - **Discover interesting features; separate sources that have been mixed together, etc. etc.**
- **Reinforcement learning (this just tries to have a good time)**
  - **Choose actions that maximise payoff**

Reinforcement learning

- **The basic paradigm of reinforcement learning is as follows: The learning agent observes an input state or input pattern, it produces an output signal [..], and then it receives a scalar "reward" or "reinforcement" feedback signal from the environment indicating how good or bad its output was.**
- **The goal of learning is to generate the optimal actions leading to maximal reward.**
- **[Tesauro ’94]**

Backgammon complexity

- **Large: 21 possible dice outcomes, for each of which about 20 legal moves.**
- **Brute force approach isn't feasible.**
- **In general we need to develop positional judgement, rather than trying to look ahead explicitly: a position is good or bad *per se*.**

TD-gammon

- **The network: simply an MLP**
- **General learning idea: the network plays against itself, and considers as a positive example a winning sequence of moves (and perhaps as a negative example a losing one).**

Strengths of TD-gammon

- **According to experts:**
- **still makes some small mistakes at tactical game, where variations can be calculated out; no wonder, it does not calculate them..**
- **is tremendous at vague positional battles, where what matters is evaluating a pattern**
- **humans have learned from it**

Unsupervised learning

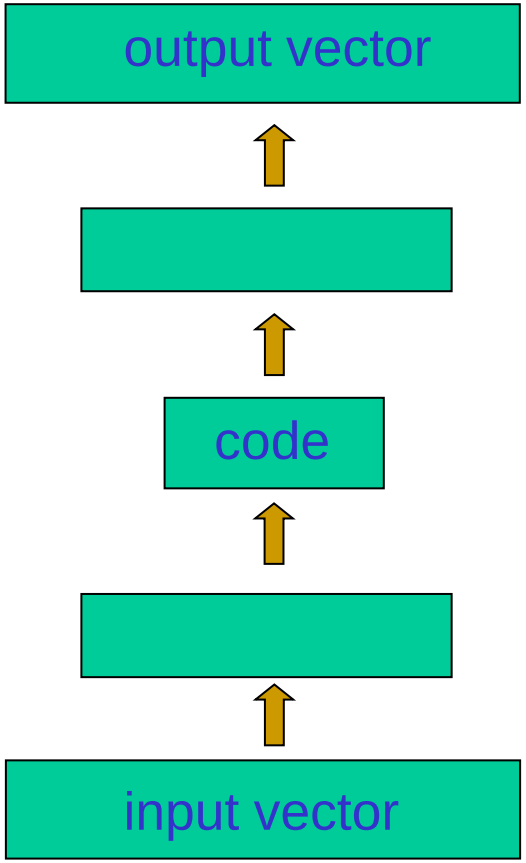
- **Without a desired output or reinforcement signal it is much less obvious what the goal is.**
- **Discover useful structure in large data sets without requiring a supervisory signal**
  - **Create representations that are better for subsequent supervised or reinforcement learning**
  - **Build a density model that can be used to:**
    - **Classify by seeing which model likes the test case data most (model selection)**
    - **Monitor a complex system by noticing improbable states.**
    - **Extract interpretable factors (causes or constraints)**
- **Improve learning speed for high-dimensional inputs**

Unsupervised learning according to the nnfaq

- **“Unsupervised learning allegedly involves no target values. In fact, for most varieties of unsupervised learning, the targets are the same as the inputs [..]. In other words, unsupervised learning usually performs the same task as an auto-associative network, compressing the information from the inputs [..].”**

Using backprop for unsupervised learning

- **Try to make the output be the same as the input in a network with a central bottleneck.**
- **The activities of the hidden units in the bottleneck form an efficient code. The bottleneck does not have room for redundant features.**
- **Good for extracting independent features**



Self-supervised backprop in a linear network

- **If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimise the squared reconstruction error.**
- **This is exactly what Principal Component Analysis does (*note: I shall break down and cry you if you spell it "principle"*).**
- **The M hidden units will span the same space as the first M principal components found by PCA**

Principal Components Analysis (PCA)

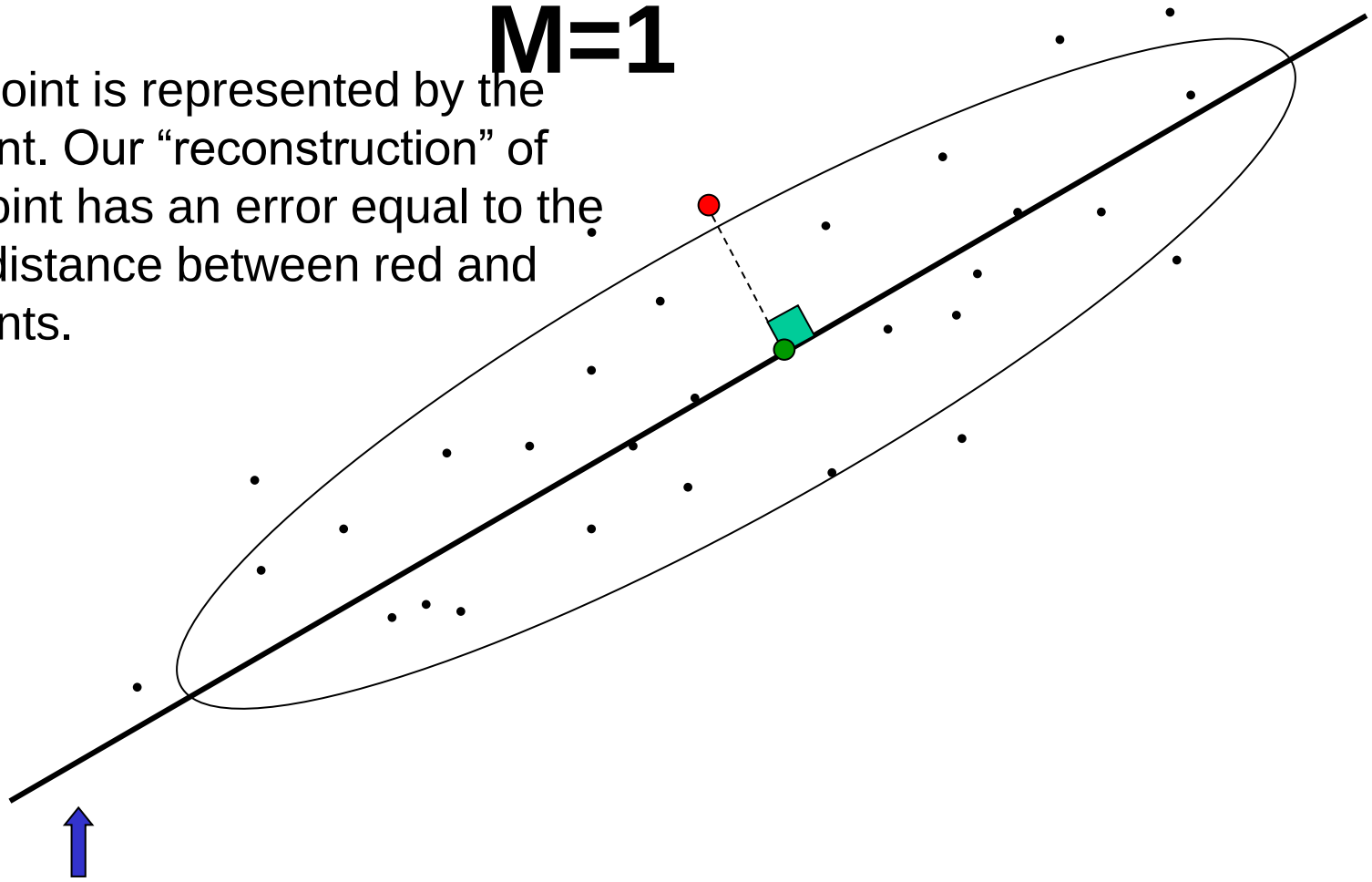
- **Takes N-dimensional data and finds the M orthogonal directions in which the data has the most variance**
- **These M principal directions form a subspace.**
- **We can represent an N-dimensional datapoint by its projections onto the M principal directions**

Principal Components Analysis (PCA)

- **PCA loses all information about where the datapoint is located in the remaining orthogonal directions.**
- **We reconstruct by using the mean value (over all the data) on the N-M directions that are not represented.**
- **The reconstruction error is the sum over all these unrepresented directions of the squared differences from the mean.**

A picture of PCA with N=2 and M=1

The red point is represented by the green point. Our “reconstruction” of the red point has an error equal to the squared distance between red and green points.



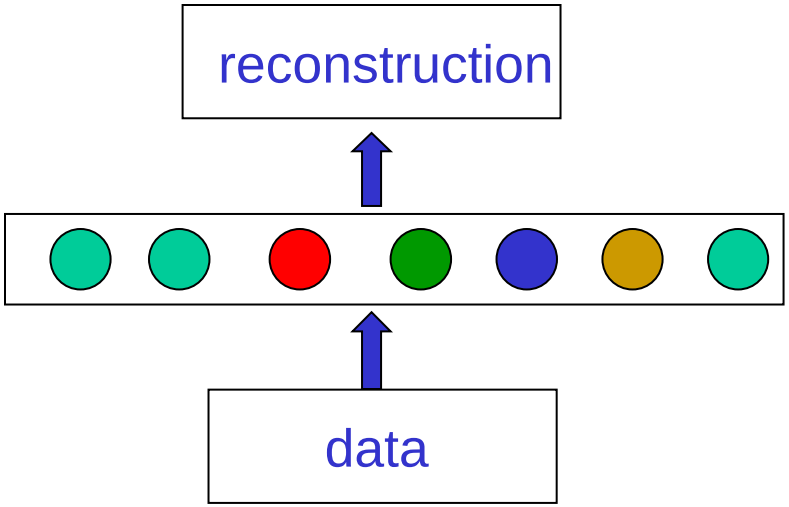
First principal component:
Direction of greatest variance

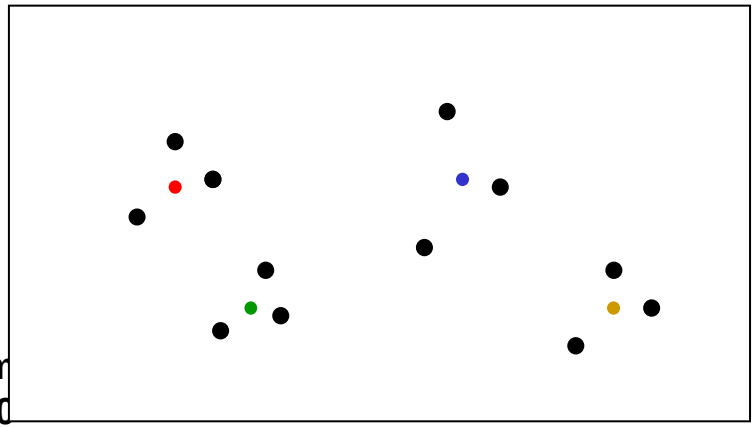
Self-supervised backprop in the non-linear case

- **Associating the data with itself (auto-associator) using a linear network is equivalent to doing Principal Component Analysis.**
- **What happens if we try to do the same using a non-linear network?**

Self-supervised backprop in the non-linear case

- If we force the hidden unit whose weight vector is closest to the input vector to have an activity of 1 and the rest to have activities of 0, we get clustering.
- The weight vector of each hidden unit (HU->output) represents the centre of a cluster.
- Input vectors are reconstructed as the nearest cluster centre.
- Number of clusters = number of HU.





Clustering and backpropagation

- We need to tie the input->hidden weights to be the same as the hidden->output weights.
- Usually, we cannot backpropagate through binary hidden units, but in this case the derivatives for the input->hidden weights all become zero
  - If the winner doesn't change – no derivative
  - The winner changes when two hidden units give exactly the same error – no derivative

Clustering and backpropagation

- **The only error-derivative is for the output weights. This derivative pulls the weight vector of the winning cluster towards the data point.**
- **When the weight vector is at the centre of gravity of a cluster, the derivatives all balance out because the c.of g. minimises squared error.**

VQ and k-means

- **These networks come in different flavours. One of the most common ones is the VQ (vector quantisation) model, by Teuvo Kohonen.**
- **Incidentally, the k-means clustering algorithm happens to be very much the same thing too.**

k-means

- 1) k-means initially assigns each cluster’s centroid to an element in the population.
- 2) It examines each component in the population and assigns it to the closest cluster.
- 3) The centroid's position is recalculated for each cluster, then we go back to 2.

Linear vs. non linear

- **Linear self-associating networks build *global* features (each one of them is based on all data points): this is a strength.**
- **Non-linear (hardmax) self-associating networks build *local* features (one cluster is essentially agnostic to the points outside it): this is a weakness.**

Linear vs. non linear (2)

- **Linear self-associating networks are *linear*. They only come up with features which are linear combinations of the inputs. This is a weakness (you can't XOR the inputs..)**
- **Non-linear self-associating networks build much stronger features.**
- **It isn't really clear what a non-hardmax self-associating network with multiple encoding and multiple decoding layers does, but it's global and non-linear (should be good!).**