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%)**

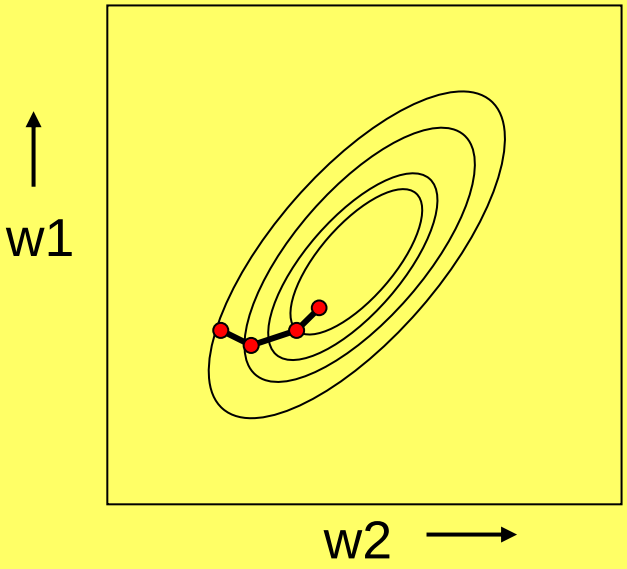
Programming assignment

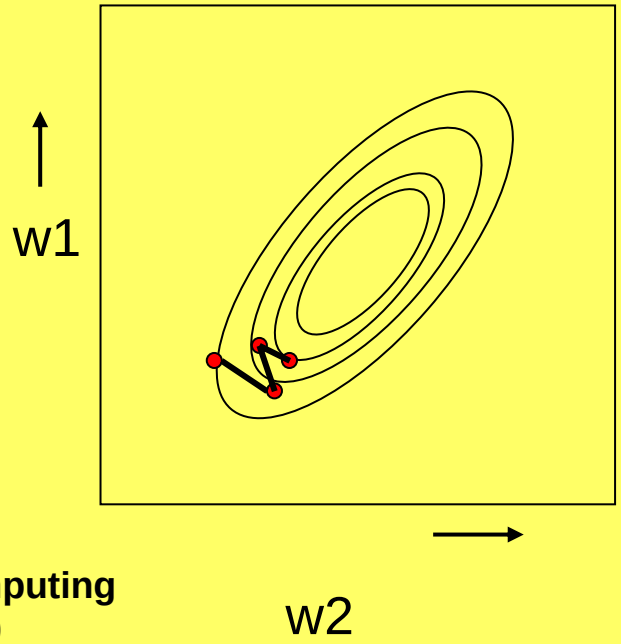
- Implement a Multi-Layer Perceptron, test it.
- The description on Brightspace.
- Submit through Brightspace code and test results by Dec the 5th at 23:59, any time zone of your choice (Baker Island?).
- 30% of the overall mark
- One third of a grade down every day late, that is: if you deserve an A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

Online versus batch learning

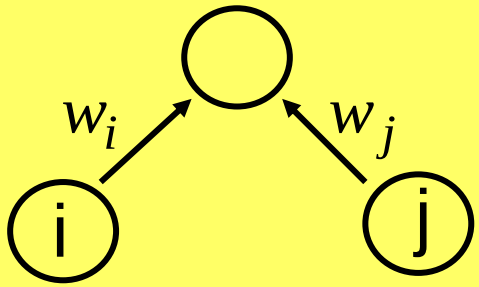
- **Batch learning does steepest descent on the error surface**

Online learning zig-zags around the direction of steepest descent.





Fixing up the error surface



Suppose that inputs i and j are highly correlated. Changing wi will change the error in the same way as changing wj.

So changing both will have larger effects on E than changing a weight that isn't correlated to the others

- **Preprocess each input so that it isn't correlated to the others**
- **The gradient will now point at the minimum**
- **This means computing the <u>covariance matrix</u> among all the outputs (OK), inverting it (*not OK*, if there are many inputs), and multiplying each input by the inverse.**

$$Cov(x_i, x_j) = \frac{1}{N}\sum_{n=1}^{N}(x_{i,n} - \mu_i)(x_{j,n} - \mu_j)$$

..fixing the gradient

- **Use an adaptive learning rate**
  - **Increase the rate slowly if it's not diverging**
  - **Decrease the rate quickly if it starts diverging**
- **Use momentum**
  - **Instead of using the gradient to change the position of the weight, use it to change the velocity of change.**
- **Use fixed step**
  - **Use gradient to decide where to go, but always go at the same pace.**

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]**

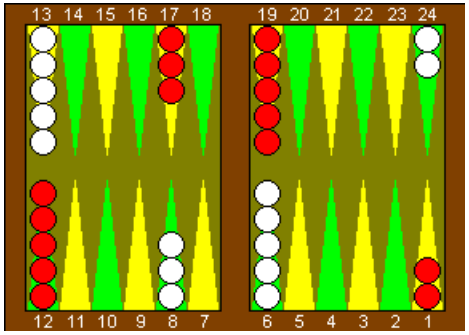
Reinforcement learning

- **In many cases the reward is also delayed (i.e., is given at the end of a long sequence of inputs and outputs). In this case the learner has to solve what is known as the "temporal credit assignment" problem (i.e., it must figure out how to apportion credit and blame to each of the various inputs and outputs leading to the ultimate final reward signal).**
- **[Tesauro ’94]**

TD-gammon

- **“A neural network that trains itself to be an *evaluation function* for the game of backgammon by playing against itself and learning from the outcome”.**
- **The appealing thing here is learning without a teacher (at least, without a full time one).**

Backgammon



- A two-player game, played on a one-dimensional track (although represented on a 2D board).
- Players take turns, roll 2 dice, move their checkers along the track based on the dice outcome.
- Win: moving all the checkers all the way to the end, and off the board.
- Gammon (double win): a player wins when the other still hasn't taken any checkers off the board.

Backgammon

- **Hitting a checker: landing on it, when it's alone; it is sent all the way back.**
- **Blocking: it is possible to build structures that make it difficult to move forward for the opponent.**

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*.**

Neurogammon

- **Previous approach by Tesauro. MLP trained in a *supervised* fashion on a training set of moves by human experts.**
- **A lot of tricks (features) included.**
- **Problem: human experts are fallible; ideas about what constitutes a good move are revised all the time.**
- **Neurogammon was a good player (best program) but far from the best humans.**

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).**

TD-gammon: inputs and outputs

- The inputs x[1], x[2], .. x[f] are the board positions during a game. They are encoded into the network in some way.
- Output y[t] is a four component vector that estimates the final outcome (judges positions): White win, Black win, White gammon, Black gammon.
- (When playing, all the moves associated with a die roll are evaluated and the best one picked.)

TD-gammon: learning

- **Update is based on the TD (time differences) rule:**

$$\Delta W_{t+1} = \eta(Y_{t+1} - Y_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w Y_k$$

- **At the final step, Y is known, so the reward is entered into the network.**

Time credit

$$\Delta W_{t+1} = \eta(Y_{t+1} - Y_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w Y_k$$

- **The parameter λ controls time credit assignment, i.e. how far ahead an action influences learning. λ=1 means that each past action is considered equally important to determine the outcome at time t. λ=0 means no memory.**

Training results

- **At the beginning the network plays randomly: very long games, not much sensible learning.**
- **Still, elementary strategies are quickly learned.**
- **Best network: 40 hidden units, 200,000 games. Roughly as good as Neurogammon.**

Adding features

- **Instead of just coding the raw configurations, encode knowledge-based features of the configuration into the network.**
- **With these features TD-gammon outperforms Neurogammon.**
- **Version 2 achieved near-parity with world class level human players.**
- **Version 3.1 has a look ahead strategy and longer training: it is unbeatable by humans.**

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**

- **“Instead of a dumb machine which can calculate things much faster than humans such as the chess playing computers, [..] a smart machine which learns from experience pretty much the same way humans do.”**