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.

Assignment 2

- **Read the paper “Finding structure in time”, by Elman (1990), up to and excluding the *Discovering the notion "word"* section.**

- **The paper is on Brightspace.**

- **Submit a 250 word MAX summary by October the 30th at 23:59 on Brightspace, any time zone of your choice (Baker Island?).**
- **7% of the overall mark**
- **1/3 of a grade down every 2 days late, that is: if you deserve an A and you’re 1-2 day late you get an A-, 3-4 days late a B+, etc.**
- **You are responsible for making sure I get it..**

Learning and gradient descent problems

- **Overfitting (general learning problem): the model memorises the examples very well but generalises poorly.**
- **GD is slow... how can we speed it up?**
  - **GD does not guarantee that the direction of maximum descent points to the minimum.**
  - **Sometimes we would like to run where it's flat and slow down when it gets too steep. GD does precisely the contrary.**
- **Local minima?**

Overfitting: formally

- A learner overfits the data if
  - It outputs a hypothesis $h(\cdot) \in H$ having true error ε and empirical error E, but
  - There is another $h'(\cdot) \in H$ having

$$E' > E \text{ and } \varepsilon' < \varepsilon$$

- In practice, during learning the error on the training examples decreases all along, but the error in generalisation reaches a minimum and then starts growing again.

Limiting the size of the model: using fewer Hidden Units

- If n=number of inputs and M=number of hidden units the VC of a MLP is proportional to (n M logM).
- Reducing M reduces the capacity of the model -> prevents overfitting.
- Of course we need to know the exact capacity we need. Too many weights: overfitting. Too few weights: underfitting.
- In practice: trial and error.

Limiting the size of the model: weight decay

- **<u>Weight-decay</u> involves adding an extra term to the cost function that penalises the squared weights.**
  - **Keeps weights small unless they have large error derivatives.**

$$C = E + \frac{\lambda}{2} \sum_i w_i^2$$

$$\frac{\partial C}{\partial w_i} = \frac{\partial E}{\partial w_i} + \lambda w_i$$

Preventing overfitting: early stopping

- If we have lots of data and a big model, it's very expensive to keep re-training it with different amounts of weight decay.
- It is much cheaper to start with very small weights and let them grow until the performance in generalisation starts getting worse.
- The capacity of the model is limited because the weights have not had time to grow big.

Sets

- **Divide the total dataset into three subsets:**
  - **Training data is used for learning the parameters of the model.**
  - **Validation data is not used for learning but is used for deciding what type of model and what amount of regularisation works best.**
  - **Test data is used to get a final, unbiased estimate of how well the network works. We expect this estimate to be worse than on the validation data.**
- **We could then re-divide the total dataset to get another (or more than one) unbiased estimate of the true error rate. N-fold Cross-Validation.**

Fighting overfitting: combining networks

- **Averaging the predictions of many different networks is a good way to avoid overfitting. *Ensemble, ensembling..***
- **It works best if the networks are as different as possible.**

Why combining networks works

- **We want to compare two expected squared errors**
  - **Method 1: Pick one of the predictors at random**
  - **Method 2: Use the average of the predictors, y**

$$\bar{y} \quad = \quad < y_i >_i \quad = \quad \frac{1}{N}\sum_{i=1}^{N} y_i$$

$$< (d - y_i)^2 >_i = < \big((d - \bar{y}) - (y_i - \bar{y})\big)^2 >_i$$

$$= < (d - \bar{y})^2 + (y_i - \bar{y})^2 - 2(d - \bar{y})(y_i - \bar{y}) >_i$$

$$= < (d - \bar{y})^2 >_i + < (y_i - \bar{y})^2 >_i \ ...$$

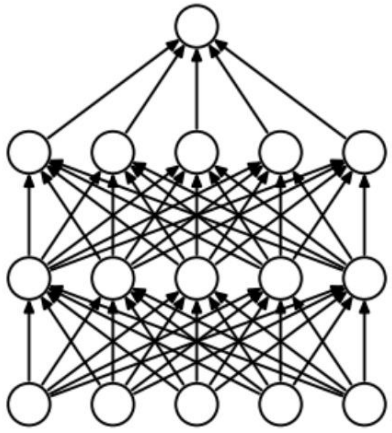
$$- 2(d - \bar{y}) < (y_i - \bar{y}) >_i$$

Why combining networks works

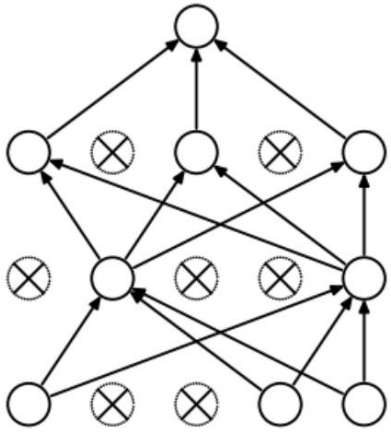
- **The average squared error of the predictors is equal to the squared error of the average of the predictors, plus the average squared difference between a predictor and the average of the predictors.**
- **In conclusion, the squared error of the average of the predictors is always smaller than the average squared error of the single predictors (unless all the predictors are exactly the same).**

Ensembling with just 1 training: dropout

- During training, at each step knock out some (different) randomly chosen connections.
- When predicting, use all connections (you need to introduce a normalising constant for this to work correctly).
- This is equivalent to having a very large ensemble of networks, but you train only once!



(a) Standard Neural Net



(b) After applying dropout.

Words of caution on dropout based on my experience

- **It doesn't always work, and there are preconditions for it to work.**
- **1) When you train with knocked out connections, you effectively have lower net capacity. So you need to have an oversized net to start with, which mitigates the gains of having fewer trainings to run. If your network has the right capacity in the first place, dropout will make it underfit.**
- **2) It does impose fairly strict constraints on what individual weights represent (reducing their cooperation), which reduces effective capacity further.**

- **That said, you typically use this in a subset of the layers. It can be a very convenient technique to avoid complex fine-tuning and model selection: just throw a humongous net at the problem, introduce dropout and wait.**

Overfitting: summary

- **Problem that occurs when the network memorises the examples but not the underlying trends.**
- **It can be prevented by:**
  - **limiting number of weights**
  - **weight decay**
  - **early stopping**
  - **combining networks (especially if they disagree)**