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%)

Learning in the Boltzmann machine with hidden units

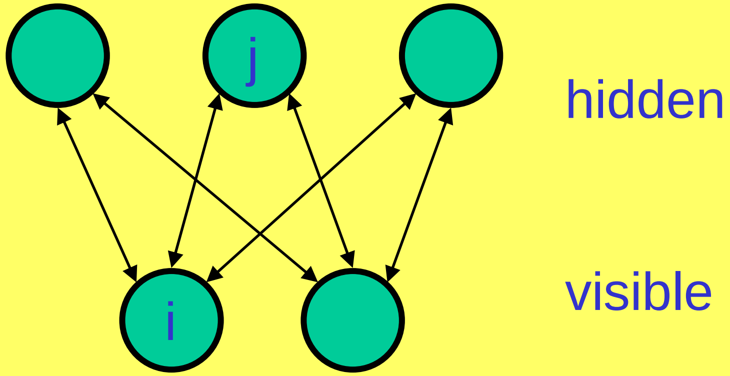
- Even in this case we have two terms in Δw_{ij}.
- The first term is the correlation between neurons i and j when the visible units are clamped to the examples.
- The second is the correlation between neurons i and j when the system is let evolve freely.

Learning in the Boltzmann machine with hidden units

- In this case both terms must be estimated by letting the network evolve many times until equilibrium, in one case with the visible units clamped, in the other freely.
- Not only: the missing units need to be estimated separately for each example.
- Can be computationally very expensive

Restricted Boltzmann Machines (RBM)

- **We restrict the connectivity to make inference and learning easier.**
  - **Only one layer of hidden units.**
  - **No connections between hidden units.**
- **It only takes one step to reach equilibrium when the visible units are clamped..**



Types of learning task

- **Supervised learning:**
  - learning to predict output when examples of input and corresponding output are available.
- **Reinforcement learning:**
  - no real supervision, only occasional payoff: e.g. I don’t know if a chess move is correct but I know if I win.
- **Unsupervised learning:**
  - I want to create an internal representation of the data, e.g. in form of clusters, extract relevant features for further tasks, etc.

Supervised learning

- A set of examples: <x,f(x)>
  - x is some object (instance) $\in \mathcal{X}$ (instance space)
  - f(·) is an *unknown* function
- Learning algorithm will guess h(·) ≈ f(·)
- Inductive learning hypothesis
  - Any h(·) that approximates f(·) well on training examples will also approximate f(·) well on new (unseen) instances x. This isn't necessarily true, of course..

Learning as refinement

- Start with a small hypothesis class (e.g., boolean conjunctions)
  – this means we need to *know a priori* something about the solution
- Use examples to infer the particular function (e.g. conjunction) in the class.

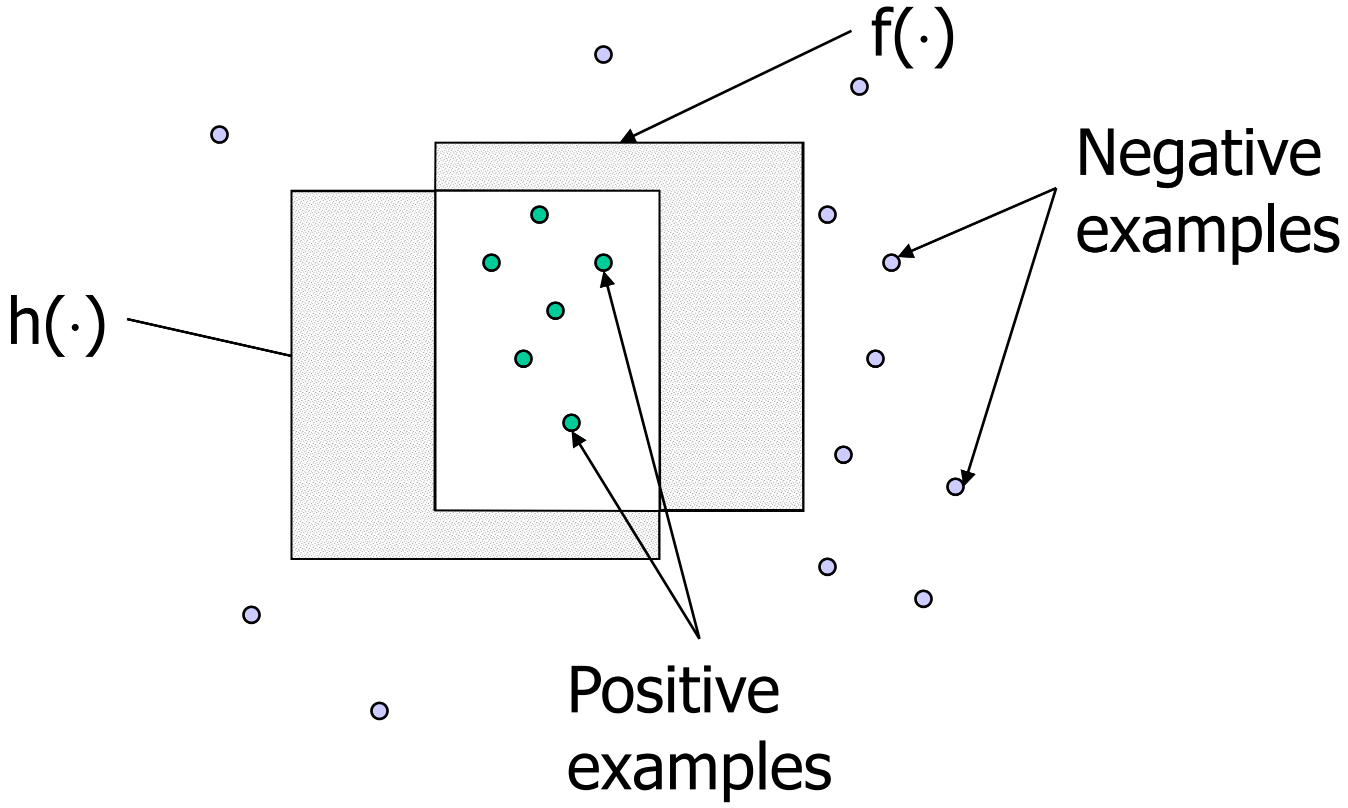
Learning as refinement

- **Start with a small hypothesis class (e.g., boolean conjunctions)**
  - **this means we need to *know a priori* something about the solution**
- **Use examples to infer the particular function (e.g. conjunction) in the class.**

Generalisation

- **So far we discussed how well we learn the training set.**
- **What really matters is how well we *generalise* i.e. how well we perform on instances not seen before.**

Generalisation



$$\text{true error} = \int_X f(x) \oplus h(x) P_D(x)\,dx$$

Generalisation

- Are there boundaries on generalisation error?
- How expressive is a model, i.e. once we know how to learn:
  - what can we learn?
  - how many examples do we need to learn?
- For instance: how complex a task can we learn with N neurons / how many examples do we need to train them?

PAC learning

- **Probably Approximately Correct (hopefully)**
- **H finite, unknown data distribution D, consistent learner**
- **Then we need at least**

$$m \geq \frac{1}{\varepsilon}(\ln|H| + \ln(1/\delta))$$

training examples to guarantee that the true error will be < ε with probability > (1-δ)

m is called *sample complexity*

Example

- **1000 functions in H, δ=1%, ε=1%**
- **$m \geq 100 (\ln 1000 + \ln 100) = 1151.3$**

Examples

Hypothesis space 1: $|H|=2^{2^n}$

$$m \geq \frac{1}{\varepsilon}(2^n \ln 2 + \ln(1/\delta))$$

Intractable!!

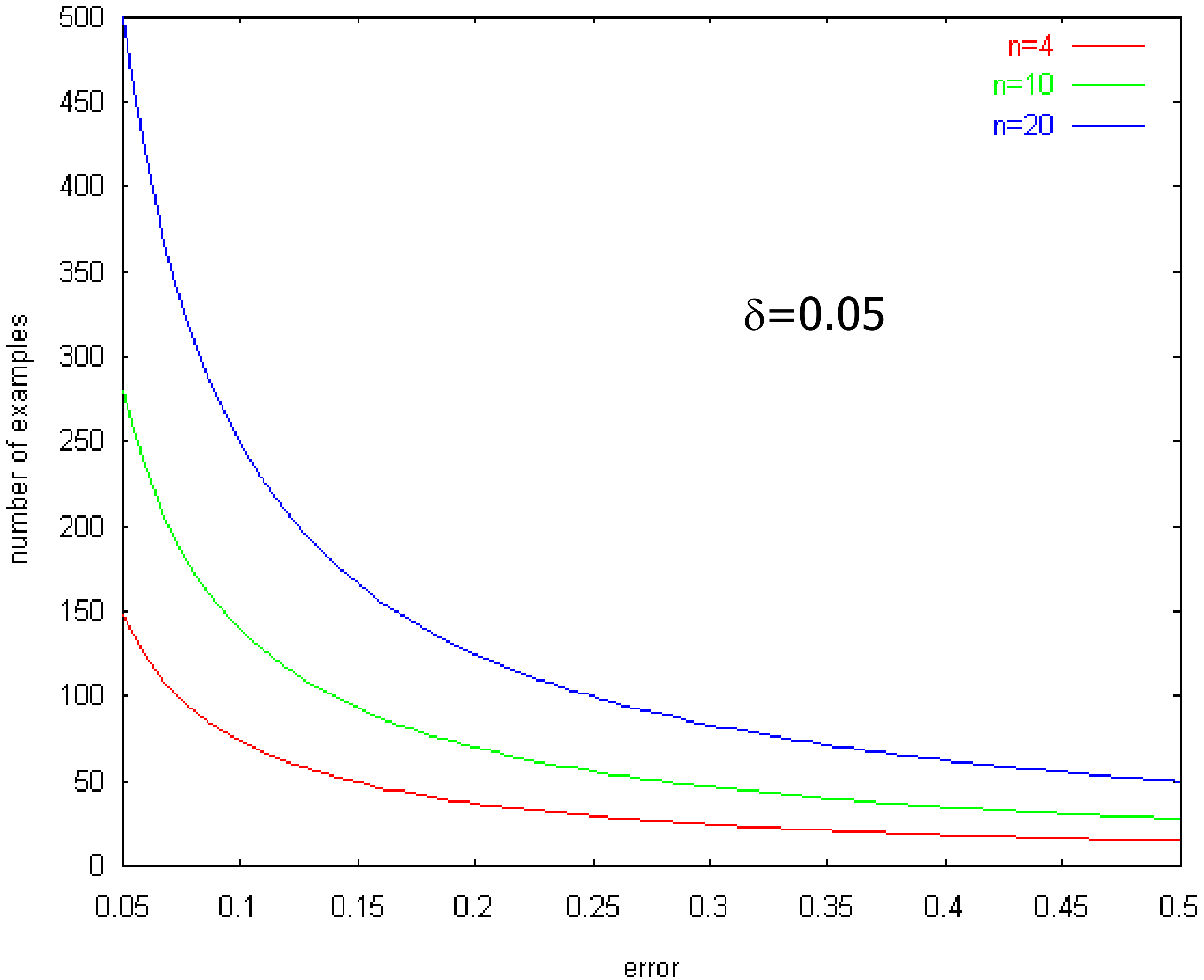
Hypothesis space 2: $|H|=3^n$

$$m \geq \frac{1}{\varepsilon}(n \ln 3 + \ln(1/\delta))$$

Polynomial

Sample complexity bound

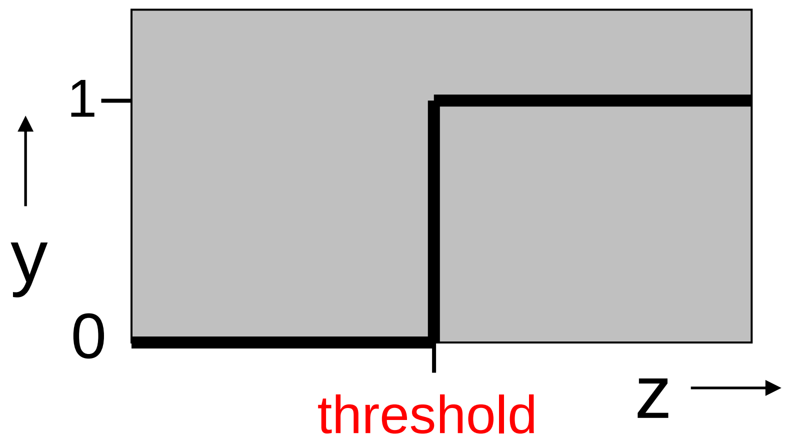
$$m \geq \frac{1}{\varepsilon}(n \ln 3 + \ln(1/\delta))$$



Binary threshold neuron

$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } z > \theta \\ 0 \text{ otherwise} \end{cases}$$



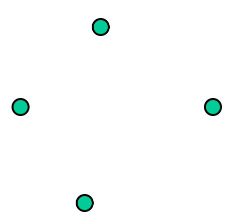
- **How powerful is it, how many examples do we need? Can we apply PAC boundaries?**

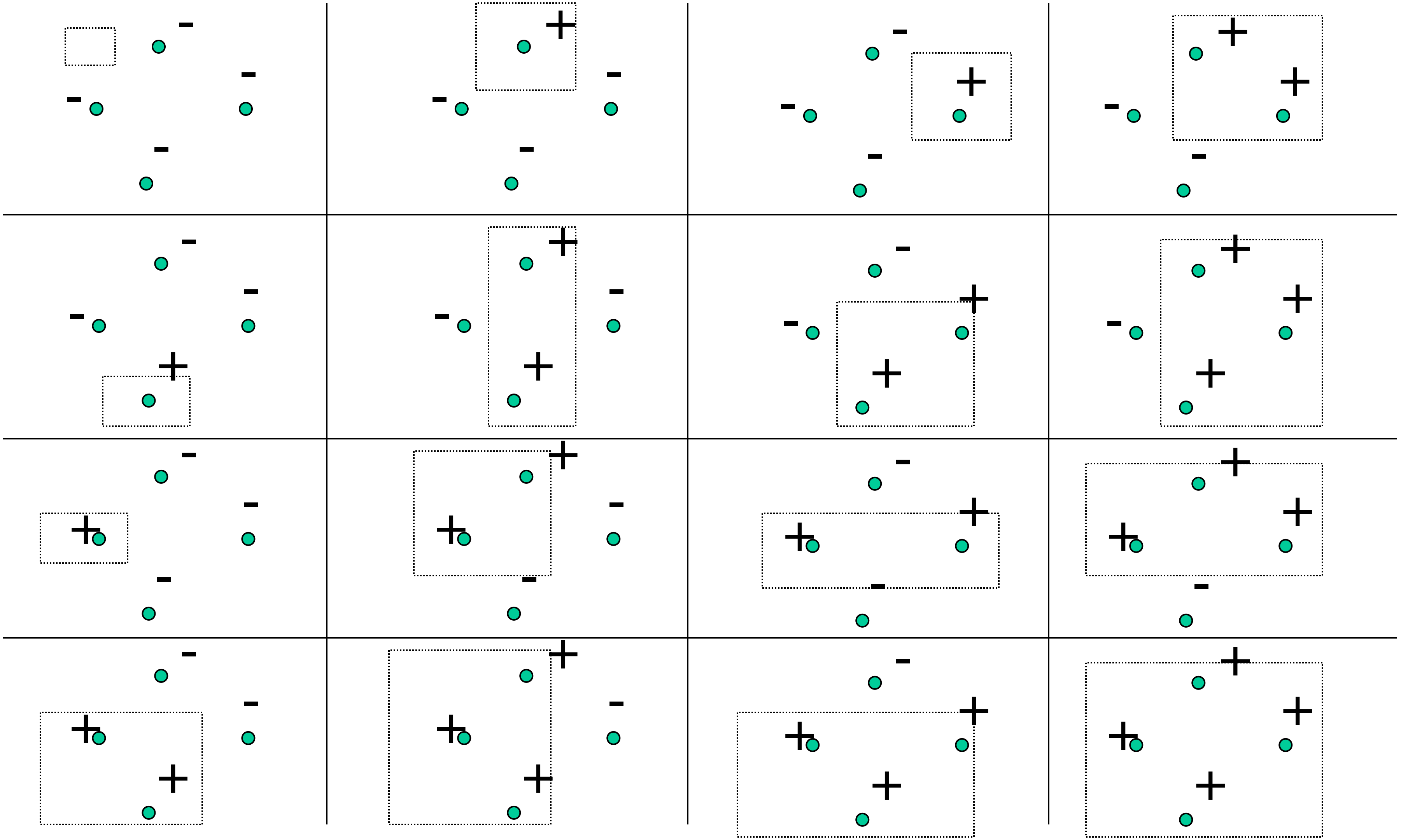
Perceptron

- **PAC works well if we have a limited space H. But for a perceptron the hypothesis space is the set of all separation hyperplanes**
- **How many hyperplanes in $\Re^n$? $|H|=\infty$ … ?**
- **Not only, but PAC is based on consistent learners. We know that perceptrons can't learn some tasks.**
- **We need an extension of PAC:**
- **VC dimension**
  V=Vapnik
  C=Chervonenkis

Shattering

- H shatters a set of points *S* if for every possible dichotomy of S (way of splitting it in 2), there exists a consistent hypothesis *h* in *H*
- Example: is this set of points shattered by the hypothesis space of all axis parallel rectangles?





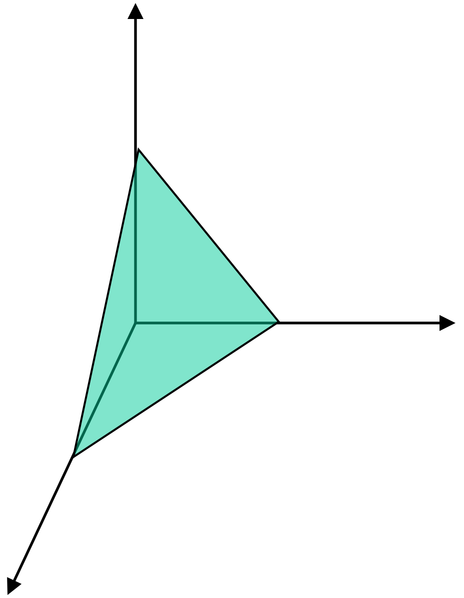
VC dimension

- VC(H) is the size of the largest set of points S that can be shattered by H

- Examples for 2D data points:
  - VC(axis-parallel-rectangles)=4
  - VC(circles)=3
  - VC(lines)=3

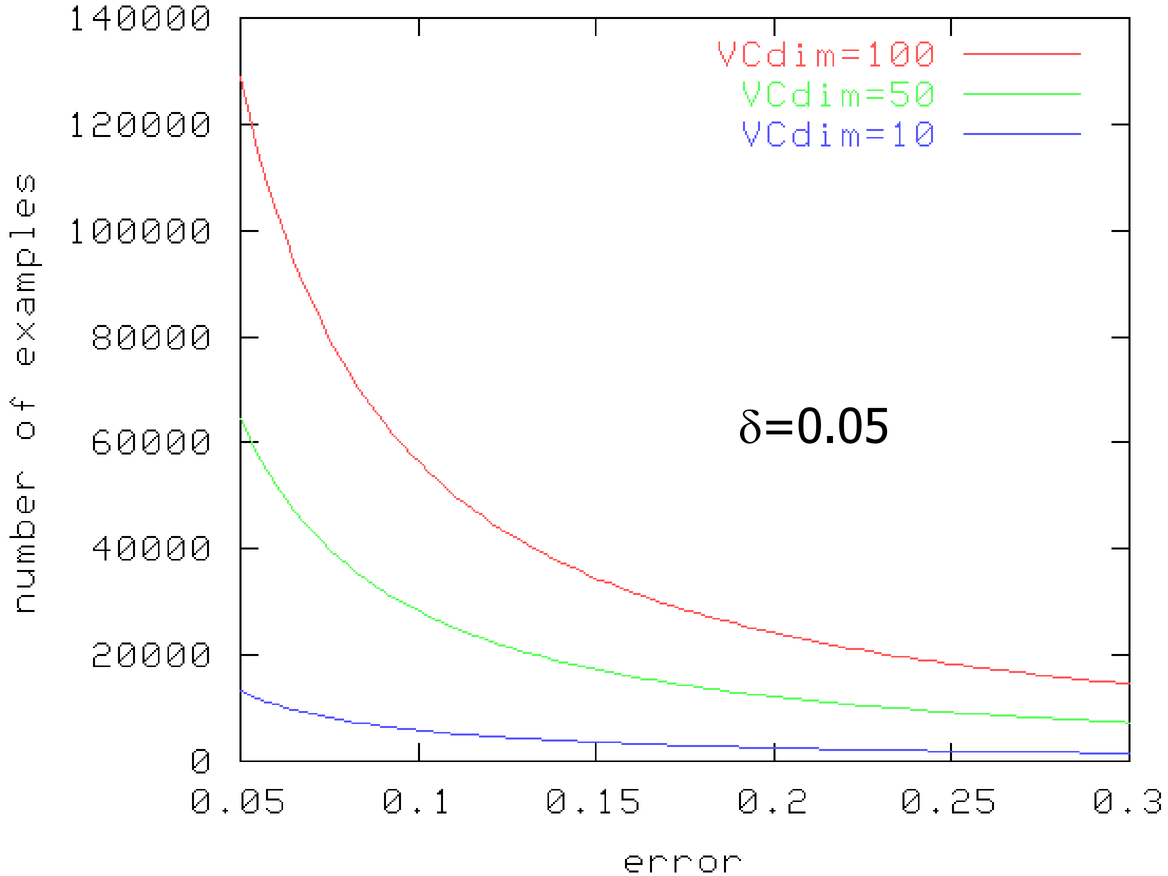
VC dimension

- For n-dimensional data points a linear decision surface has VC(H) = n+1
- For instance, for a perceptron with n inputs (n-dimensional data points), VC(H)=n+1



PAC learning and VC dim

$$m \geq \frac{1}{\varepsilon}\left(4\log_2(2/\delta) + 8VC(H)\log_2(13/\varepsilon)\right)$$



PAC learning for a perceptron

- n inputs -> VC(H)=n+1

- Example:
- 100 inputs, δ=1%, ε=1%
- $m \geq 100\ (4 \log 200 + 8\ 101 \log 1300) = 8388.8$

- **Enough about learning theory for the moment.**
- **We will get back to it soon to compute the expressive power of a new kind of network..**