Generalization

- A warm and fuzzy discussion of supervised learning and generalization.
- A formal definition of generalization.
- A formal definition of learnability.
- Worst case learning and the PAC formalism.
- Maximizing expected generalization (the Bayesian approach.)
- Information theoretic approaches.
- (Unsupervised learning.)

Barak Pearlmutter

Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

What generalization means to me

To generalize means that you see some examples of something, you figure out what's going on, and you then apply whatever it is you figured out to cases you see in the future.

We will consider this in the context of a supervised learning task. You get some questions annotated with correct answers. Later you use what you learned to answer some new questions.

Definition: the amount of generalization is how well you do in your attempts to answer these testing questions.

Issues in the definition of generalization:

- Who makes up the questions?
- Who figures out the right answers?
- What about errors in the sample answers? Or even the questions?
- How does the scoring work? E.g. partial credit for ambiguous answers?

Barak Pearlmutter

Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

What are we trying to learn?

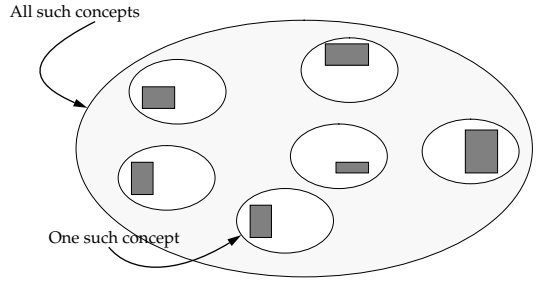
We will consider only "yes or no" answers: i.e. *dichotomies* or *classifications*.

Each such dichotomy is a rule that gives a yes/no classification for each possible question. The possible questions are called *inputs* and the rule is called a *concept*.

Concepts can also be considered as functions from the input space to $\{0, 1\}$. Or most intuitively, as a subset of the input space: that subset which is mapped to 1.

The *concept class* is the set of all the rules that we will be allowed to consider.

For instance:



Concept Class: axis aligned rectangles, $\Re^2 \rightarrow \{0, 1\}$

This is a "four dimensional" space of concepts, because each concept can be expressed as four numbers, the (x, y) coordinates of the upper left and lower right corners of the rectangle.

Barak Pearlmutter

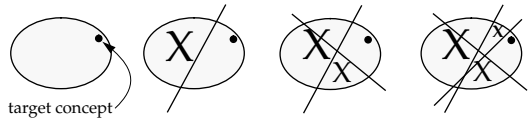
Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

How to use the training data

Let's maintain a set of all the members of the concept class consistent with the examples we've seen so far. Before we've seen any training cases at all, this is the entire concept class. As each training case becomes available, we throw away any concepts inconsistent with it.



After a while new cases will tend to be consistent with all the remaining concepts. This is generalization.

Issues:

- How long will this take? How many training cases are necessary in order to have good odds of getting most of the test cases right?
- Which of the remaining concepts should we use for guessing on the test cases? Or could we somehow combine the remaining concepts?
- How should we actually compute this? Maybe the set is too big to explicitly maintain.

It seems pretty clear that, in any case, the smaller the concept class we start out with, the quicker generalization will start. Most of the remainder of this lecture consists of formalizing whatever notion of size we mean *smaller* with respect to.

But for now, let us consider a simple implication of this "smaller is better" idea.

Barak Pearlmutter

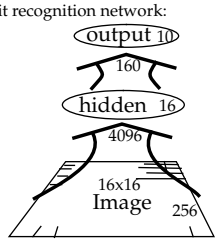
Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

Application to neural networks: weight equivalences

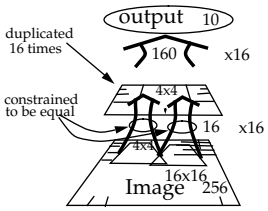
Consider this large digit recognition network:



- 4000+ weights, freedom to do lots of silly things. Big concept class.
- Spacial structure not known to network, will have to extract this structure (translation and scale invariance, say) from the exemplars.

By cutting down on the number of weights, we can constrain the network, making the concept class smaller. But this reduces the computational power.

Another possibility is to constrain certain weights to be equal, and/or give units local receptive fields. Here, down to 3000+ weights. This reduces concept class size and *increases* the computational power.



Barak Pearlmutter

Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

Who makes up the questions?

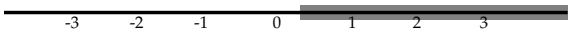
All formal theories of generalization share one crucial idea: the sample inputs *(exemplars)*, are drawn from a probability distribution over the space of inputs. The test cases are drawn from *the same distribution*.

This definition finesses the problem of trying to generalize to a situation you can't get any information about, and is used in most formal theories concerning generalization.

It is also a pretty reasonable assumption with real-world data.

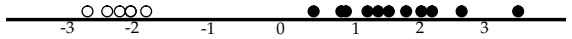
Example:

Concept class: 1D threshold functions, target function shown.

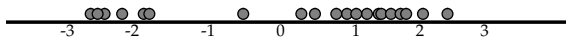


Input sampling distribution: Gaussian + uniform

Exemplars:



Test cases:



Notice how only a couple test cases fell in a region where their correct classification can't be inferred from the training data.

Barak Pearlmutter

Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

Cross validation

Since the training and testing data are drawn from the same distribution, one popular approach is to reserve some of the training data for a *check set*, train on the rest, then use the check set to estimate the generalization rate.

Advantages:

- Easy to do.
- Works well in practice.
- Allows you to fiddle with learning parameters and network architecture for maximum generalization.
- Incremental version is cheap and simple.

Disadvantages:

- Takes away potential training data.
- Extra computation.
- What if the estimate is noisy? We might end up making important decisions randomly.
- If you fiddle with the learning parameters too much, you can end up with a gross overestimate of the generalization rate; information from the check set contaminates the learning regime and architecture.

Barak Pearlmutter

Department of Psychology

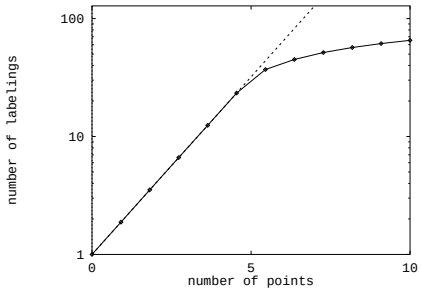
P.O. Box 11A Yale Station

New Haven, CT 06520-7447

The graph Tony wanted me to show you

Consider making a graph for a particular concept class, in which the x-axis is the number of input samples drawn, and the y-axis (drawn with a log scale) is the number of different ways concepts from the concept class can label these points. Since the number of possible labelings might depend on the particular set of input points chosen, consider the maximum for any set of input points of the given size.

Such a graph will start with a straight line (on our log scale) reflecting a region where we can label our n points in any way we like, namely any of the 2^n conceivable labelings. Then a transition will occur, and our freedom will be diminished. For reasons beyond the present scope, after this transition, no matter what the concept class, the curve must flatten out from exponential to polynomial!



The location of this transition point is called the VC dimension of the concept class.

Barak Pearlmutter

Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

Who gets to choose? (a) your worst enemy

Let's say that the concept class is known, but an *adversary* gets to choose the target concept, and also the input distribution.

It turns out we can now figure out how many exemplars are necessary to *Probably* (odds of at least $1-\delta$) get a concept which is *Approximately* ($1-\varepsilon$ of the time) *Correct*. If this number is polynomial in $1/\varepsilon$ and $1/\delta$ then the concept class is called *PAC learnable.*

It can be shown that a concept class is PAC learnable if and only if it has a finite *Vapnik-Chervonenkis dimension*, a quantity we shall define on the next slide. $P(P(TestCaseCorrect) > 1-\varepsilon) > 1-\delta$ is guaranteed if you have either $8\varepsilon^{-1}Dim(C)\log 8\varepsilon^{-1}Dim(C)$ or $4\varepsilon^{-1}\log 2\delta^{-1}$ exemplars, whichever is greater. This holds no matter which element of the concept class consistent with the data the learner guesses.

Example: For aligned rectangles, how many exemplars are needed to be 80% sure, no matter what that input distribution and target concept, that we'll get a concept which classifies 95% of the test cases correctly. $Dim(Rect^2) = 4$ so we need

$$max\left(\frac{4}{0.05}\log\frac{2}{0.20}, \frac{(8)(4)}{0.05}\log\frac{(8)(4)}{0.05}\right) = max(185, 4136) = 4136$$

Summary: If you have a

- *known* concept class of *finite* VC dimension,
- then even if the target concept and input distribution are chosen *maliciously,*
- the number of exemplars needed is a slowly growing function of the VC dimension of the concept class and the reciprocal of the required generalization error rate.

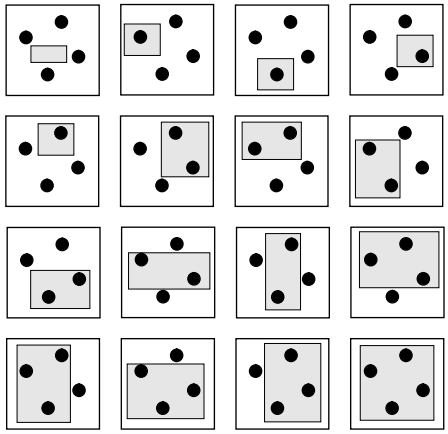
Barak Pearlmutter
Department of Psychology
P.O. Box 11A Yale Station
New Haven, CT 06520-7447

The Vapnik-Chervonenkis dimension

The VC dimension of a concept class is a combinatorial property of that class: the maximum number of input points that can be chosen such that the chosen points are *shattered* by the concept class.

A set of input points is shattered by a concept class if each subset of that set of inputs can be picked out by some concept in the concept class.

Example: the VC dimension of aligned rectangles is *four* because we can pick out any subset of the four points shown with an aligned rectangle but



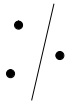
we can't find five points such that we can thus pick out any subset.

Barak Pearlmutter
Department of Psychology
P.O. Box 11A Yale Station
New Haven, CT 06520-7447

Another example

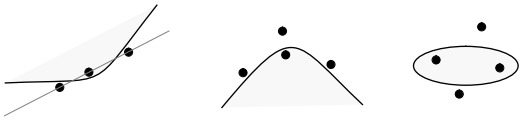
Let us calculate the VC dimension of a perceptron with two inputs.

It is easy to exhibit three points that can be shattered: three points at the virtex of a triangle can be shattered, since it is easy to pick out any 0, 1, 2 or 3 of them with a half plane. Thus the VC dimension is at least three.



To show it to be exactly three, we must show that no four points can be shattered. Let's try to arrange four points so that they can be shattered.

If three points are colinear then they can't be shattered, as it would be impossible to pick out the middle point without getting one of the end points. Therefore of the four points to be shattered, no three can be colinear. This leaves two possibilities: three of the points can form a



triangle containing the fourth, or the four points can form the vertices of a convex quadrilateral. In the triangle case, the center point can not be picked out without getting one of the other points, and in the quadrilateral case, two diagonally opposite points can not be picked out without getting one of the other two points.

Therefore the VC dimension of a half-planes in $\Re^2$ is three.

Barak Pearlmutter
Department of Psychology
P.O. Box 11A Yale Station
New Haven, CT 06520-7447

Intuitive properties of the VC dimension

- *Usually* corresponds to the number of free parameters, plus or minus 1.
- A boolean combination of n concept classes $C_1, \ldots, C_n$ has VC dimension $C_1 + \cdots + C_n$.
- Intuitive relationship to generalization: points of absolute freedom.
- Works the same for whether the input space is continuous or discrete.
- Fun to calculate. At least for me.

Exercise: Calculate the VC dimension of the following concept classes.

- Half-line, as in the "Who makes up the questions?" viewgraph.
- Triangles in $\Re^2$.
- Unoriented rectangles in $\Re^2$ not necessarily oriented along the axes.
- Circles in $\Re^2$.
- Spheres in $\Re^3$.
- Linear discrimination in $\Re^n$.

Note: As we've seen, a VC dimension calculation has two parts:

1. exhibit a set of n input points that can be shattered.
2. prove that no set of $n+1$ input points can be shattered.

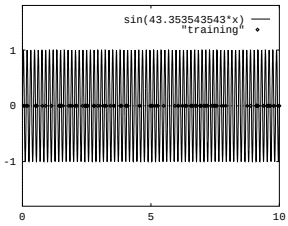
As we've also seen, usually the first is easier.

Barak Pearlmutter
Department of Psychology
P.O. Box 11A Yale Station
New Haven, CT 06520-7447

Counterintuitive properties of the VC dimension

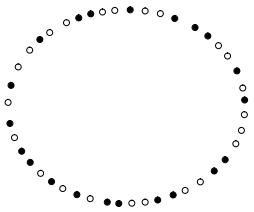
- Usually corresponds to number of free parameters: multilayer perceptrons, polynomials, polytopes, etc. *But not always!*

Example: the VC dimension of the one-dimensional concept class $sign(\sin\Omega x)$ is infinite!



- Some seemingly innocuous concepts have high VC dimensionality.

Example: Convex regions in the plane have infinite VC dimension!



Barak Pearlmutter
Department of Psychology
P.O. Box 11A Yale Station
New Haven, CT 06520-7447

Application to neural networks

Using VC dimension arguments, it has been shown (Baum and Haussler) that if you train a multilayer network with n weights on $O(\varepsilon^{-1}n\log\varepsilon^{-1}n)$ exemplars to an error rate of less than ε then the generalization error rate will almost certainly be less than 2ε.

Unfortunately:

- The constant factor is unknown. (Actually, known to be less than 47 and conjectured to be 1.)
- The log factor does not seem to appear in practice.
- Experience on some tasks shows generalization with less data than the formula calls for, even with the log factor removed and a constant factor of 1.

Barak Pearlmutter
Department of Psychology
P.O. Box 11A Yale Station
New Haven, CT 06520-7447

Who gets to choose? (b) randomly chosen

There is a big problem with the preceding worst-case framework: much better results are achieved in practice. In particular, the log factor does not seem to appear in the real world.

Perhaps we should look at the *average case* instead. **Issue:** average over what? Only possible answer: over some *prior distribution:*

- There is a known probability distribution over the concept class, and the target concept is chosen according to that distribution.

This is known as the *Bayesian* approach.

Intuitively: As before, we use exemplars to rule out untenable hypotheses. But now we either chose the most likely remaining hypothesis, or average over the remaining hypotheses according to their *posterior* distribution.

Note: Sometimes the distribution from which inputs are chosen is assumed to be "nice," i.e. not too sharp. This corresponds to no infinite precision, or some finite noise level, both of which are very reasonable real-world assumptions. Under this assumption, Haussler showed that the VC dimension of convex sets becomes finite, with the exact number depending on how much noise there is in the inputs.

Note: Sometimes the distribution over the inputs is assumed to be known to the learner. This is typically necessary for optimality.

Barak Pearlmutter
Department of Psychology
P.O. Box 11A Yale Station
New Haven, CT 06520-7447

How Reverend Bayes stacks up

Advantages:

- *Optimality:* averaging over the posterior is the best possible strategy.
- If you can't do that, choosing the most likely, or even just sampling the posterior, can be shown to be not much worse.
- If the priors are close enough to being right, the log factor goes away.
- The "best prior" is known: $P(C) \propto 2^{Kol(C)}$ where $Kol(C)$ is the *Kolmogorov complexity* of concept C, the length of the shortest computer program (in an agreed upon language) that computes C. This formalizes *Occam's Razor*: that the simplest theory is to be preferred.
- Many priors approximate the Kolmogorov complexity under certain assumption, e.g. Rissanen's *Minimum Description Length* or *MDL* rule.
- Handles noisy training data and infinite VC dimension concept classes very naturally.
- Can apply mathematics from statistical mechanics, so it gives physicists jobs that don't involve saying "care for some fries with that?"

Disadvantages:

- Kolmogorov complexity is *uncomputable!*
- You are forced to come up with a prior distribution.
- All the usable priors have knobs to tweak, so it can't be really cookbook.
- People argue all the time about the "right priors," but in practice it seems to be less of a problem than one might at first suppose.
- Computational costs can be prohibitive.
- Performance and required exemplars depends on the priors, whose correctness is difficult to test.
- Cutting edge research.

Barak Pearlmutter
Department of Psychology
P.O. Box 11A Yale Station
New Haven, CT 06520-7447

Application to neural networks

In a Bayesian context, we express the probability of a hypothesis as being composed of two terms:

- How well the hypothesis matches the data.
- The hypothesis's a-priori likelihood.

If we assume that the errors of a network's outputs are Gaussian distributed, and that the weights of that network are also Gaussian distributed, we get a measure whose negative log is

$$E = \underbrace{\alpha \sum_{p} (y_p - d_p)^2}_{E_{data}} + \beta \sum_{i,j} w_{ij}^2$$

where α and β are the standard deviations, the first term is the usual backpropagation squared error, and the second term gives *weight decay:* each weight is decayed towards zero each time step, according to its magnitude.

Weight decay was first used by Pearlmutter to improve generalization and keep the weights small, before the Bayesian rationale was known. Weight decay has another desirable property: after training, each weight's magnitude is proportional to its importance, as $w_{ij} \propto -\partial E_{data} / \partial w_{ij}$. This makes interpreting the weights much easier.

- Weight decay should be used when the weights are to be examined after training, or if training data is scarce.
- Other sorts of weight decay result from non-Gaussian assumptions.
- The Bayesian approach also suggests *ensemble networks*, in which multiple networks are trained and their outputs averaged, which can give up to a $\sqrt{2}$ improvement in expected generalization.

Barak Pearlmutter

Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

An information theoretic approach

Another way of looking at generalization involves *information theory.*

Consider the amount of information each exemplar gives us, and consider the amount of information that can be stored in your neural network. Once you reach the network's capacity, generalization starts.

Maxim: generalization starts where memorization ends.

Going back to our old picture, with each exemplar cutting off a chunk of no longer tenable hypotheses, try the following strategy:

- Measure (in bits) the amount of information each exemplar gives us. For dichotomies, that's one bit per exemplar.
- Measure (in bits) the capacity of the concept class to store information. This is just $-\log|C|$. Durbin showed that for linear threshold networks with one hidden layer and one output unit, this comes to exactly 2 bits per weight.
- 2 bits per weight is a good rule of thumb for backpropagation nets of arbitrary configuration, barring bottlenecks. Four bits per weight tops, except with exceptional training effort on artificial problems.
- *Make sure the network has few enough weights that the training set can't just be memorized*. With a single binary output, this comes to at least twice as many training cases as weights.

This extends naturally to extra output units: one bit per output per exemplar. It also extends gracefully to learning real-valued functions, because you always have an idea of the accuracy of the numbers, and therefore of the amount of information in each one.

Note: remember weight equivalences? Well, 2 weights constrained to be equal only count as 1 weight here.

Barak Pearlmutter

Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

Concluding remarks

- One complaint registered against using neural networks on real world problems is that they are a black art: choosing the number of hidden units, architecture, etc. is not a cookbook process.

  This is no longer true. We have seen that sometimes it *is* a cookbook process, and that there are simple rules of thumb or straightforward procedures for estimating the such things without too much thought. Just like applying Fourier transforms or fitting polynomials.
- Despite all the theory, experimentation and trial-and-error remain the best tools for achieving good generalization.
- Since the whole point of using automatic learning procedures is generalization, there is furious activity in this subfield. I have only touched upon a few highlights here.

- There are branches of the study of generalization and learning that seem irrelevant to neural networks as currently conceived, such as formal learning theory, which deals with learning formal languages. Such mathematical theories are of interest in their own right, and also have application to psychology. And hopefully one day to neural networks.

Barak Pearlmutter

Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

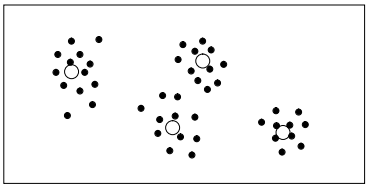
Unsupervised learning

We can cast *unsupervised* learning tasks into a similar framework, in which there is a performance criterion, the expected generalization rate, and we fiddle around trying to make them do their job properly, with a precise definition of what we mean by properly.

Instead of trying to figure out what the proper output is for each input, in an unsupervised learning task we try to estimate the distribution from which the inputs are drawn.

By having a network map each input to a "probability," somehow encoded, learning becomes maximization of the probability of the exemplars, and better generalization becomes maximizing the probability of the test cases.

Example: competitive learning.



The open circles correspond to the units' weight vectors, and the small black dots are exemplars. We formalize the task by saying that the inputs are generated by randomly picking a unit and perturbing its weight vector with Gaussian noise. The objective is to maximize the probability of the given inputs. Taking log probabilities gives squared distance minimization, but with *soft* rather than *hard* competition. This procedure has been shown superior in practice, e.g. for modems on noisy telephone lines.

Barak Pearlmutter

Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447

- Unsupervised learning can be put into the same framework as supervised learning. It need not be a wish-washy warm-and-fuzzy branch of neural networks. We can actually say what an unsupervised learning algorithms *should* do, check if it's really doing it, and modify the learning algorithms to make it perform better. This is a marked advance over other approaches to unsupervised learning.

Barak Pearlmutter

Department of Psychology

P.O. Box 11A Yale Station

New Haven, CT 06520-7447