LING5702: Lecture Notes 21
A Model of Generalization (Learning)

Contents

Language skills allow us to predict meanings from utterances.

These predictions are generalized to allow us to learn new words.

These predictions may be learned like other kinds of generalized predictions, through repetition.

- prey come out when it rains
- predators don't come out when it's dark
- when a stick is in an anthill, ants crawl up it
- words don't begin with *nd*
- subjects come before predicates

21.1 Difference between memory and generalization/learning

Brains learn to predict things (prey/predator behavior, word segmentaion/order) through repetition.

We already covered how cued associations form by firing of pre-synaptic and post-synaptic neurons.

But that just explains memory:

- Lowly Worm came of his burrow Sep 26, 2007, during a light shower.
- Mommy said hand me a shovel.

It does not explain how **generalizations** are formed:

- Prey come out when it rains.
- words don't begin with *nd*.

21.2 Overview of learning mechanism

Generalizations can be learned by individual neurons through repeated refinement of predictions.

- **remember** the kind of events you didn't predict or just barely did (esp. borderline cases)
  (there is some evidence that such surprisals are stored in associative memory)
- **replay** these events over and over in your mind:
  - start with any initial synaptic weights
  - proportionately increase synaptic weights of active pre-synaptic neurons for false negatives
  - proportionately decrease synaptic weights of active pre-synaptic neurons for false positives
- seems to happen during sleep [Diekelmann and Born, 2010].

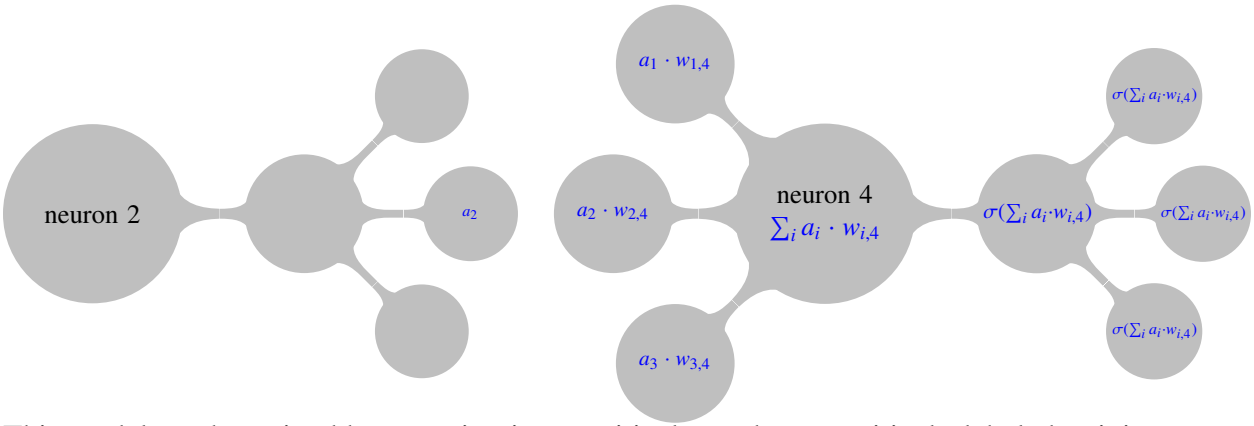
This is essentially logistic regression.

Specifically, we will assume this kind of learning is used to define certain features of mental states.
(For example, 'type' features of elementary predications, used for classifying events.)

21.3 Training process

Remember our model of neural activation (where $\sigma(x) = \frac{1}{1+e^{-x}}$):

$$a_j = \sigma\left(\sum_i a_i \cdot w_{i,j}\right)$$



This model can be trained by exposing it to positively- and non-positively-labeled training examples (in particular, labeled by ones and zeros). In human language learning, positively-labeled examples might represent instances where words are found to break at a particular position or not to break at that position. Training proceeds by initializing the weights $w_{1,j}, \ldots, w_{I,j}$ (typically to random values), then iterating over the training examples and modifying the weights to improve the fit of the prediction to labeled values of the training examples. At each iteration, these weights are adjusted upward or downward based on the difference between the labeled values of the training examples and the predicted value a_j of the neural model, given the values of

the independent variables (features, or pre-synaptic activations) in the training examples. Given $w^{(0)}_{1,j\ldots I,j} \in \mathbb{R}^I$ and $\mathcal{D} \subseteq \mathbb{R}^{I+1}$, the updated weight $w^{(t)}_{i,j}$ after each iteration t is:

$$w^{(t)}_{i,j} = w^{(t-1)}_{i,j} + \sum_{\langle a_1,\ldots,a_I,a_j\rangle \in \mathcal{D}} \underbrace{a_i \cdot \underbrace{\underbrace{\left(a_j - \underbrace{\sigma\left(\sum_{i'} a_{i'} \cdot w^{(t-1)}_{i',j}\right)}_{\text{prediction}}\right)}_{\text{error}}}_{\text{error} \times \text{pre-synaptic activation}}} \tag{1}$$

21.4 Example

Here is an example training process.

Suppose we encounter six phone sequences, some of which lead to yummy cookies:

/kʊkitɑim/	a cookie is forthcoming
/ɑimgəntəkʊkərəʊst/	no cookie is forthcoming
/wɑnðəməŋki/	no cookie is forthcoming
/wɑntəkʊki/	a cookie is forthcoming
/ɑilkʊkɪtəθɪnɪt/	no cookie is forthcoming
/wɑnðədəŋki/	no cookie is forthcoming

and we have pre-synaptic neurons active for phonemes, and a post-synaptic neuron for YUMMY!:

pre-synaptic				post-synaptic
a_1(/tə/)	a_2(/ki/)	a_3(/kʊ/)	a_4(/wɑn/)	a_j(YUMMY!)
0.00	1.00	1.00	0.00	1.00
1.00	0.00	1.00	0.00	0.00
0.00	1.00	0.00	1.00	0.00
1.00	1.00	1.00	1.00	1.00
1.00	0.00	1.00	0.00	0.00
0.00	1.00	0.00	1.00	0.00

We'll also assume a weighted threshold input $w_{0,j}$ (generally inhibitory).

Intuitively, we should expect to find that a_3(/kʊ/) and a_2(/ki/) are good indicators of YUMMY.

Start with random initial weights: $w^{(0)}_{0,j\ldots 4,j} = 0.69, 0.79, 0.20, 0.09, -0.06$.

Then make predictions and increase synaptic weights of active pre-synaptic neurons by the error...

pre-synaptic activation				post-synaptic	prediction	error	error × pre-synaptic activation			
a_1	a_2	a_3	a_4	a_j	$\sigma(..)$	$a_j - \sigma(..)$	a_1	a_2	a_3	a_4
0.00	1.00	1.00	0.00	1.00	0.73	0.27	0.00	0.27	0.27	0.00
1.00	0.00	1.00	0.00	0.00	0.83	-0.83	-0.83	-0.00	-0.83	-0.00
0.00	1.00	0.00	1.00	0.00	0.70	-0.70	-0.00	-0.70	-0.00	-0.70
1.00	1.00	1.00	1.00	1.00	0.85	0.15	0.15	0.15	0.15	0.15
1.00	0.00	1.00	0.00	0.00	0.83	-0.83	-0.83	-0.00	-0.83	-0.00
0.00	1.00	0.00	1.00	0.00	0.70	-0.70	-0.00	-0.70	-0.00	-0.70
						-2.62	-1.50	-0.97	-1.23	-1.24
					+ $w^{(0)} = w^{(1)}$:	-1.93	-0.71	-0.77	-1.14	-1.30

And again:

pre-synaptic activation				post-synaptic	prediction	error	error × pre-synaptic activation			
a_1	a_2	a_3	a_4	a_j	$\sigma(..)$	$a_j - \sigma(..)$	a_1	a_2	a_3	a_4
0.00	1.00	1.00	0.00	1.00	0.02	0.98	0.00	0.98	0.98	0.00
1.00	0.00	1.00	0.00	0.00	0.02	-0.02	-0.02	-0.00	-0.02	-0.00
0.00	1.00	0.00	1.00	0.00	0.02	-0.02	-0.00	-0.02	-0.00	-0.02
1.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
1.00	0.00	1.00	0.00	0.00	0.02	-0.02	-0.02	-0.00	-0.02	-0.00
0.00	1.00	0.00	1.00	0.00	0.02	-0.02	-0.00	-0.02	-0.00	-0.02
						1.90	0.95	1.94	1.93	0.96
					+ $w^{(1)} = w^{(2)}$:	-0.04	0.24	1.17	0.79	-0.34

And again:

pre-synaptic activation				post-synaptic	prediction	error	error × pre-synaptic activation			
a_1	a_2	a_3	a_4	a_j	$\sigma(..)$	$a_j - \sigma(..)$	a_1	a_2	a_3	a_4
0.00	1.00	1.00	0.00	1.00	0.87	0.13	0.00	0.13	0.13	0.00
1.00	0.00	1.00	0.00	0.00	0.73	-0.73	-0.73	-0.00	-0.73	-0.00
0.00	1.00	0.00	1.00	0.00	0.69	-0.69	-0.00	-0.69	-0.00	-0.69
1.00	1.00	1.00	1.00	1.00	0.86	0.14	0.14	0.14	0.14	0.14
1.00	0.00	1.00	0.00	0.00	0.73	-0.73	-0.73	-0.00	-0.73	-0.00
0.00	1.00	0.00	1.00	0.00	0.69	-0.69	-0.00	-0.69	-0.00	-0.69
						-2.57	-1.32	-1.11	-1.20	-1.24
					+ $w^{(2)} = w^{(3)}$:	-2.61	-1.08	0.06	-0.40	-1.58

And again:

pre-synaptic activation				post-synaptic	prediction	error	error × pre-synaptic activation			
a_1	a_2	a_3	a_4	a_j	$\sigma(..)$	$a_j - \sigma(..)$	a_1	a_2	a_3	a_4
0.00	1.00	1.00	0.00	1.00	0.05	0.95	0.00	0.95	0.95	0.00
1.00	0.00	1.00	0.00	0.00	0.02	-0.02	-0.02	-0.00	-0.02	-0.00
0.00	1.00	0.00	1.00	0.00	0.02	-0.02	-0.00	-0.02	-0.00	-0.02
1.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
1.00	0.00	1.00	0.00	0.00	0.02	-0.02	-0.02	-0.00	-0.02	-0.00
0.00	1.00	0.00	1.00	0.00	0.02	-0.02	-0.00	-0.02	-0.00	-0.02
						1.88	0.96	1.92	1.91	0.96
					$+ w^{(3)} = w^{(4)}$:	-0.73	-0.12	1.97	1.51	-0.61

After 16 more iterations, the predictions are generally correct...

pre-synaptic activation				post-synaptic	prediction	error	error × pre-synaptic activation			
a_1	a_2	a_3	a_4	a_j	$\sigma(..)$	$a_j - \sigma(..)$	a_1	a_2	a_3	a_4
0.00	1.00	1.00	0.00	1.00	0.93	0.07	0.00	0.07	0.07	0.00
1.00	0.00	1.00	0.00	0.00	0.07	-0.07	-0.07	-0.00	-0.07	-0.00
0.00	1.00	0.00	1.00	0.00	0.07	-0.07	-0.00	-0.07	-0.00	-0.07
1.00	1.00	1.00	1.00	1.00	0.61	0.39	0.39	0.39	0.39	0.39
1.00	0.00	1.00	0.00	0.00	0.07	-0.07	-0.07	-0.00	-0.07	-0.00
0.00	1.00	0.00	1.00	0.00	0.07	-0.07	-0.00	-0.07	-0.00	-0.07
						0.20	0.26	0.34	0.34	0.26
					$+ w^{(19)} = w^{(20)}$:	-5.56	-0.56	4.74	4.26	-1.04

...and the weights are concentrated on a_3(/kʊ/) and a_2(/ki/), matching our intuitions.

Now we can **generalize** to predict responses for stimuli not in our training set:

- /gɛtəbæibi/ : $\frac{1}{1+e^{-(1\cdot-0.56+0\cdot-4.74+0\cdot4.26+0\cdot-1.04)}} = 0.3635$ – not yummy,
- /huwɑntsəkʊki/ : $\frac{1}{1+e^{-(0\cdot-0.56+1\cdot-4.74+1\cdot4.26+0\cdot-1.04)}} = 0.9998$ – YUMMY!

Note the prediction has the form of a **conditional probability** – a basis for probabilistic reasoning.

21.5 What about complex predictors?

This model only allows individual antecedent neurons to be predictors.

Sometimes we may want combinations of predictors (rain and humidity, phoneme sequences).

Use model of cortical neurons as non-linear 'convolution' function (dimensionality expansion):

- liquid state machine / reservoir computing [Maass et al., 2002]
- echo state network

21.6 Details of Optimization

Why does this update work?

It's because the derivative (slope) of weights fades to zero as the probability is maximized:

$$
\begin{aligned}
&\frac{\partial}{\partial \mathbf{W}_{[j,i]}} \overbrace{\frac{1}{N} \sum_{\langle y,\mathbf{a}\rangle \in \mathcal{D}}}^{\text{expectation}} \overbrace{-\ln \overbrace{\frac{e^{\delta_y^\top \mathbf{W}\,\mathbf{a}}}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{a}}}}^{\mathsf{P}(y|\mathbf{a})}}^{\text{loss}} \\
&= \frac{1}{N} \sum_{\langle y,\mathbf{a}\rangle \in \mathcal{D}} \frac{\partial}{\partial \mathbf{W}_{[j,i]}} - \ln \frac{e^{\delta_y^\top \mathbf{W}\,\mathbf{a}}}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{a}}} && \text{sum rule} \\
&= \frac{1}{N} \sum_{\langle y,\mathbf{a}\rangle \in \mathcal{D}} \frac{\partial}{\partial \mathbf{W}_{[j,i]}} - \left(\ln e^{\delta_y^\top \mathbf{W}\,\mathbf{a}} - \ln \sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{a}} \right) && \text{log of fraction} \\
&= \frac{1}{N} \sum_{\langle y,\mathbf{a}\rangle \in \mathcal{D}} \frac{\partial}{\partial \mathbf{W}_{[j,i]}} - \delta_y^\top \mathbf{W}\,\mathbf{a} + \ln \sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{a}} && \text{log of exponentiation} \\
&= \frac{1}{N} \sum_{\langle y,\mathbf{a}\rangle \in \mathcal{D}} -[\![y{=}j]\!]\, \mathbf{a}^\top \delta_i + \frac{\partial}{\partial \mathbf{W}_{[j,i]}} \ln \sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{a}} && \text{sum,product rule} \\
&= \frac{1}{N} \sum_{\langle y,\mathbf{a}\rangle \in \mathcal{D}} -[\![y{=}j]\!]\, \mathbf{a}^\top \delta_i + \left(\frac{1}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{a}}} \right) \frac{\partial}{\partial \mathbf{W}_{[j,i]}} \sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{a}} && \text{derivative of log} \\
&= \frac{1}{N} \sum_{\langle y,\mathbf{a}\rangle \in \mathcal{D}} -[\![y{=}j]\!]\, \mathbf{a}^\top \delta_i + \left(\frac{e^{\delta_j^\top \mathbf{W}\,\mathbf{a}}}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{a}}} \right) \frac{\partial}{\partial \mathbf{W}_{[j,i]}} \delta_j^\top \mathbf{W}\,\mathbf{a} && \text{sum,product rule} \\
&= \frac{1}{N} \sum_{\langle y,\mathbf{a}\rangle \in \mathcal{D}} -[\![y{=}j]\!]\, \mathbf{a}^\top \delta_i + \left(\frac{e^{\delta_j^\top \mathbf{W}\,\mathbf{a}}}{\sum_{y'} e^{\delta_{y'}^\top \mathbf{W}\,\mathbf{a}}} \right) (\mathbf{a}^\top \delta_i) && \text{product rule} \\
&= -\delta_j^\top \frac{1}{N} \underbrace{\left(\mathbf{Y}^\top - \overbrace{\exp(\mathbf{W}\,\mathbf{A}^\top)}^{\text{numerator}} \overbrace{\operatorname{diag}\left(\mathbf{1}^\top \exp(\mathbf{W}\,\mathbf{A}^\top) \right)^{-1}}^{\text{denominator}} \right)}_{\text{error}} \underbrace{\mathbf{A}}_{\text{prediction}}\, \delta_i && \text{def. of inner product}
\end{aligned}
$$

Start with any $\mathbf{W}^{(0)}$; move in opposite direction of error times active inputs $\mathbf{A}$, to reduce loss:

$$
\mathbf{W}^{(t)} = \mathbf{W}^{(t-1)} + \frac{1}{N} \underbrace{\left(\mathbf{Y}^\top - \exp\left(\mathbf{W}^{(t-1)}\, \mathbf{A}^\top \right) \operatorname{diag}\left(\mathbf{1}^\top \exp\left(\mathbf{W}^{(t-1)}\, \mathbf{A}^\top \right) \right)^{-1} \right)}_{\text{error}} \mathbf{A}
$$

This algorithm is called **gradient descent**.

This generalization is a **softmax** – it optimizes over several mutually-exclusive neural detectors.

It sometimes gets simplified into a **rectified linear unit** ('ReLU'), which just uses a zero/one slope.

References

[Diekelmann and Born, 2010] Diekelmann, S. and Born, J. (2010). The memory function of sleep. *Nature Reviews Neuroscience*, 11:114–126.

[Maass et al., 2002] Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560.