Lecture 11: Discriminative vs. Bayesian Classification

Mark Hasegawa-Johnson

ECE 417: Multimedia Signal Processing, Fall 2020

Outline

Review: Neural Network

Let's review how a neural net works. Suppose we have a net with two layers, whose weight matrices are

$$W^{(1)} = \left[\begin{array}{cc} 2 & -1 \\ 0 & 1 \end{array}\right], \quad W^{(2)} = \left[\begin{array}{cc} -0.1 & 0.03 \\ 0.2 & 0.05 \end{array}\right]$$

and with no bias vectors. Suppose it uses ReLU nonlinearity in the hidden layer.

As in Faster-RCNN, let's suppose that the two different outputs are treated in two different ways:

1. The first element of the output vector, $\hat{y}_0$, is a regression output: no output nonlinearity. Loss function is MSE if the classification target is 1 ($y_1 = 1$), otherwise the loss is zero.
2. The second element of the output vector, $\hat{y}_1$, is a classification output: sigmoid nonlinearity, scored with binary cross entropy.

Neural Network Example

Let's suppose we have a minibatch with just one training token:

$$\vec{x} = [4, 1]$$

Suppose that the regression target (the first output target) is $y_0 = 0.4$, and the classification target (the second output target) is $y_1 = 0$ (i.e., no object is present), so that

$$\vec{y} = [0.4, 0]$$

Forward Pass: First layer

Suppose the input vector is $\vec{x} = [4, 1]$. The hidden layer excitation is

$$\vec{e}^{(1)} = \vec{x}W^{(1)} = [4, 1]\begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} = [8, -3]$$

The hidden layer activation is

$$\vec{h} = \text{ReLU}\left(\vec{e}^{(1)}\right) = [8, 0]$$

Forward Pass: Second Layer

The second layer excitation is

$$\vec{e}^{(2)} = \vec{h}W^{(2)} = [8, 0]\begin{bmatrix} -0.1 & 0.03 \\ 0.2 & 0.05 \end{bmatrix} = [-0.8, 0.24]$$

The activation function is linear for the first output, but sigmoid for the second output:

$$\hat{y} = \left[-0.8, \frac{1}{1+e^{-0.24}}\right] = [-0.8, 0.56]$$

Loss Function

Remember that the target is $\vec{y} = [0.4, 0]$, and the network output is $\hat{y} = [-0.8, 0.56]$. The first output, $\hat{y}_0$, is scored using mean squared error if $y_1 = 1$, otherwise it is not scored at all, thus

$$\mathcal{L}_r = y_1 \frac{1}{2}(y_0 - \hat{y}_0)^2 = 0 \times \frac{1}{2}(-0.8 - 0.4)^2 = 0 \times 0.72 = 0$$

The second output, $\hat{y}_1$, is scored using binary cross entropy, thus

$$\mathcal{L}_c = -\left(y_1 \ln \hat{y}_1 + (1 - y_1) \ln(1 - \hat{y}_1)\right) = -\ln(1 - 0.56)$$

Backward Pass: Second Layer

Both BCE and MSE have the same simple form for the output-layer loss gradient:

$$\nabla_{\vec{e}^{(2)}}\mathcal{L} = (\hat{y} - \vec{y})$$

But the first term (the regression loss) is scored if and only if $y_1 = 1$. Since, in our example, $y_1 = 0$, we have

$$\nabla_{\vec{e}^{(2)}}\mathcal{L} = [0 \times (-0.8 - 0.4), (0.56 - 0)] = [0, 0.56]$$

Backward Pass: First Layer Activations

The derivative of loss with respect to the first-layer activations is obtained by back-propagating from the second-layer, which is just multiplying by the transpose of the weight matrix:

$$\nabla_{\vec{h}}\mathcal{L} = \nabla_{\vec{e}^{(2)}}\mathcal{L} W^{(2),T}.$$

In our example,

$$\nabla_{\vec{h}}\mathcal{L} = [0, 0.56] \left[\begin{array}{cc} -0.1 & 0.2 \\ 0.03 & 0.05 \end{array} \right] = [0.0168, 0.028]$$

Backward Pass: First Layer Excitations

The first-layer excitation gradient is is obtained by multiplying the activation gradient by the derivative of the nonlinearity.

$$\nabla_{\vec{e}^{(1)}}\mathcal{L} = \nabla_{\vec{h}}\mathcal{L} \odot \frac{\partial h}{\partial e^{(1)}}$$

In the case of ReLU, the derivative is either 0 or 1, so

$$\nabla_{\vec{e}^{(1)}}\mathcal{L} = [0.0168, 0.028] \odot [1, 0] = [0.0168, 0]$$

Weight Gradients

The weight gradients are the vector outer products of the forward pass and the backward pass:

$$\nabla_{W^{(1)}}\mathcal{L} = (\vec{x})^T (\nabla_{\vec{e}^{(1)}}\mathcal{L}) = \begin{bmatrix} 4 \\ 1 \end{bmatrix} [0.0168, 0] = \begin{bmatrix} 0.0672 & 0 \\ 0.0168 & 0 \end{bmatrix}$$

$$\nabla_{W^{(2)}}\mathcal{L} = \left(\vec{h}\right)^T (\nabla_{\vec{e}^{(2)}}\mathcal{L}) = \begin{bmatrix} 8 \\ 0 \end{bmatrix} [0, 0.56] = \begin{bmatrix} 0 & 4.48 \\ 0 & 0 \end{bmatrix}$$

Summary

- Forward-pass is a matrix multiply:

$$\vec{e} = \vec{h}W$$

- Backward-pass is multiplication by the transposed matrix:

$$\nabla_{\vec{h}}\mathcal{L} = (\nabla_{\vec{e}}\mathcal{L})\,(W)^T$$

- Weight gradient is a vector outer product:

$$\nabla_W\mathcal{L} = \left(\vec{h}\right)^T(\nabla_{\vec{e}}\mathcal{L})$$

Outline

Speech Recognition: So Far

Speech recognition using a nearest-neighbor classifier

- . . . works very well for isolated-word recognition, in vocabularies of up to about ten different words.
- . . . fails. . .
  - . . . for detection/segmentation. Nearest-neighbors can't tell you where the word started, where it ended.
  - . . . for continuous speech recognition. Nearest-neighbors can't transcribe a sequence of words.
  - . . . for large vocabularies. If you want to add a new word to the vocabulary, you need to record examples of that word; not very scalable, if you want 100k words.

Large-Vocabulary Continuous Speech Recognition (LVCSR)

An LVCSR has two components:

1. Acoustic model. This is a neural net that classifies which speech sound is being produced at any given instant.
2. Pronunciation model + Language model. Converts a sequence of speech sounds to a sequence of words. Three technologies, each requires more training data than the last:
   - hidden Markov model (HMM)
   - recurrent neural network (RNN)
   - attention-based sequence-to-sequence (Transformer)

Acoustic Event Detection, Video Event Detection

BTW, the same two parts exist in most acoustic event detection (AED) and multimedia activity transcription models:

1. Acoustic/Visual model. This is a neural net that classifies which acoustic event/visual event is occurring at any given instant.
2. Sequence model: arranges atomic acoustic/visual events into acoustic scenes or complex events/activity sequences.

Large-Vocabulary Continuous Speech Recognition (LVCSR)

Today we'll focus on this one:

- Acoustic model. This is a neural net that classifies which speech sound is being produced at any given instant.

Thursday we'll focus on this one:

- Pronunciation model + Language model. Converts a sequence of speech sounds to a sequence of words:
  - hidden Markov model (HMM)

Outline

Large-Vocabulary Continuous Speech Recognition (LVCSR)

Today we'll focus on this one:

- Acoustic model. This is a neural net that classifies which speech sound is being produced at any given instant.

But what does that mean, "which speech sound is being produced"?

Phonemes

A phoneme is

- a speech segment (temporally contiguous) that
- can be used to make up new words, but is also used in existing words, and
- if you change it to a different phoneme, you can change the meaning of the word.

Phonemes: Example

Who'd heed a gardener who had never hid his head in a hood? How'd you believe him? Have you heard that he hayed, or hoed, or carried a hod, or hied his hoy to HUD?

General American English has 15 vowels, if you count schwa ([ə])

Example	IPA	ARPA	Example	IPA	ARPA	Example	IPA	ARPA
heed	[i]	IY	who'd	[u]	UW	heard	[ɝ]	ER
hid	[ɪ]	IH	hood	[ʊ]	UH	how'd	[aʊ]	AW
hayed	[e]	EY	hoed	[o]	OW	hied	[ɑɪ]	AY
head	[ɛ]	EH	HUD	[ʌ]	AH	hoy	[ɔɪ]	OY
had	[æ]	AE	hod	[ɑ]	AA	a	[ə]	AX

Phoneme Notations

There are two types of phoneme notations that you should know about for this course.

1. The International Phonetic Alphabet (IPA: `https://en.wikipedia.org/wiki/International_Phonetic_Alphabet`). Invented in Europe around 1888.
2. ARPABET (`https://en.wikipedia.org/wiki/ARPABET`) is a set of ASCII (plaintext) codes for English phonemes. Still used on systems where unicode support is uncertain.

Phonemes: Example

The quick brown fox jumped over the lazy dog.
[ðə kwɪk bɹaʊn fɑks dʒʌmpt ovɝ ðə lezi dɑg]

General American English has 24 consonants

Stops & Affricates	IPA	ARPA	Fricatives	IPA	ARPA	Nasals& Glides	IPA	ARPA
poe	[p]	P	fan	[f]	F	moo	[m]	M
bo	[b]	B	van	[v]	V	no	[n]	N
tow	[t]	T	thin	[θ]	TH	sing	[ŋ]	NG
dough	[d]	D	than	[ð]	DH	woe	[w]	W
cho	[tʃ]	CH	Sue	[s]	S	low	[l]	L
joe	[dʒ]	JH	zoo	[z]	Z	row	[ɹ]	R
ko	[k]	K	ship	[ʃ]	SH	yo	[j]	Y
go	[g]	G	beige	[ʒ]	ZH	ho	[h]	HH

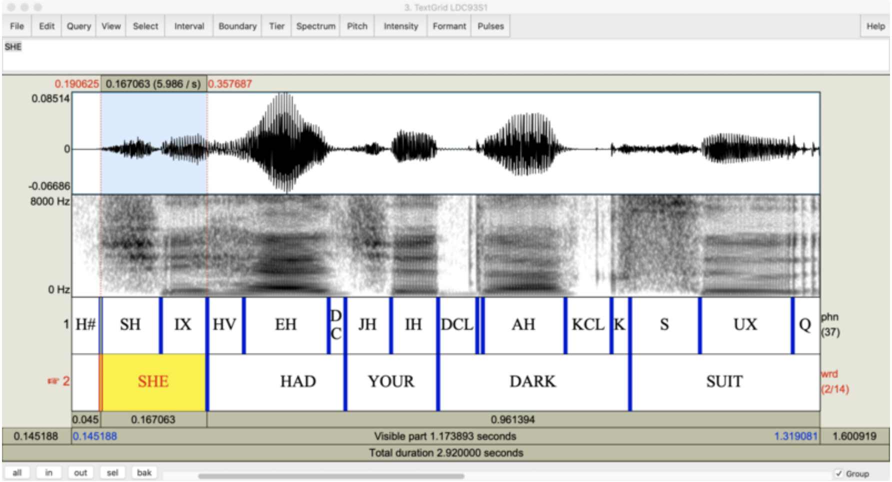
How to use phonemes

Phonemes are used:

1. ...for detection/segmentation. Nearest-neighbors can't tell you where the word started, where it ended.
2. ...for continuous speech recognition. Nearest-neighbors can't transcribe a sequence of words.
3. ...for large vocabularies. If you want to add a new word to the vocabulary, you need to record examples of that word; not very scalable, if you want 100k words.

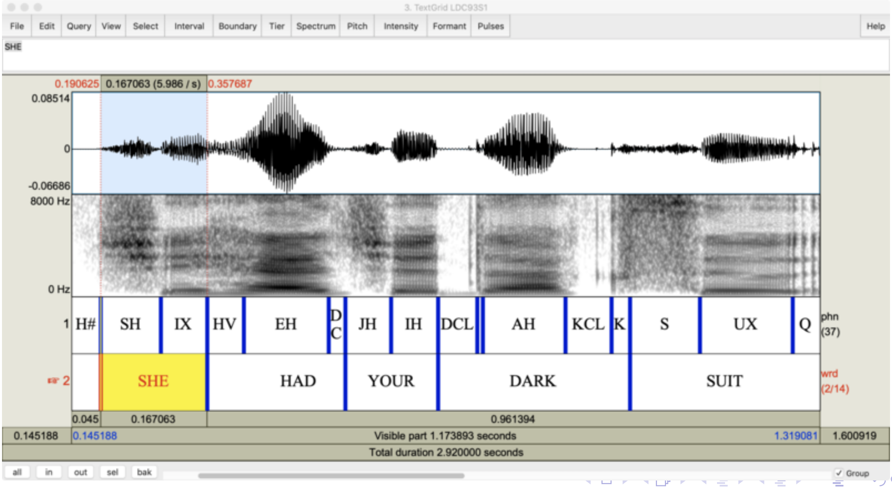
How to Use Phonemes, #1: Segmentation

Suppose you want to find where each word starts and ends. Phoneme models can help:



How to Use Phonemes, #2: CSR

Suppose you want to transcribe a sequence of words. You can do that by trying to recognize a sequence of phones, restricted to only those sequences that form valid sentences:



How to Use Phonemes, #2: Large Vocabulary

Suppose you have good models of each phoneme. Now you want to recognize the word "supercalifragilisticexpialidocious." No problem. You just create a new word model, by stringing together the phoneme models like this:

Dictionary entry for "supercalifragilisticexpialidocious"

supɝkælɪfrædʒɪlɪstɪkɛkspiælɪdoʃʌs

Generalizing from one language to another

- Two **phonemes** are different if they distinguish two words, e.g., "bat" vs. "pat." Therefore, **phonemes are language-dependent**.
- However, many languages have similar phonemes, e.g., most languages have /mama/.
- **Phones** are discrete segmental units, like phonemes, but not required to be language-dependent. In fact, we mostly just choose a phone set which is most convenient for our own software.

Phones vs. Phonemes: Example

Two phones that are different phonemes in French, but the same phoneme in English

- French has the words **ou** ([u], "or") and **eu** ([y], "had"), that sound like the same vowel in English, but are different vowels in French.
- The phone /y/ (rounded /i/) is not part of the phoneme inventory of English. If an English speaker hears it, they think you're saying either /u/ or /i/.

Two phones that are different phonemes in English, but the same phoneme in Spanish

English has the words **thin** (/θɪn/) and **sin** (/sɪn/). In Spanish, these sound like two versions of the same word (**cien**), pronounced with European vs. Latin American accent, respectively.

"Language Independent and Language Adaptive Large Vocabulary Speech Recognition," Schultz & Waibel, 1998

Phonemes [Worldbet]	KO	SP	CR	TU	JA	$\sum$
n,m,s,l,tS,p,b,t,d,g,k	X	X	X	X	X	
i,e,o	X	X	X	X	X	14
f,j,z		X	X	X	X	
r,u	X	X	X	X		
dZ	X		X	X	X	6
a	X	X	X			
S			X	X	X	
h	X			X	X	
4	X	X			X	4
$\tilde{n}$,x,L		X	X			
A				X	X	
N	X	X				
V,Z			X	X		
y,7	X			X		
ts			X		X	10
p',t',k',dZ',s',oE,oa,4i,	X					
uE,E,$\wedge$,i$\wedge$,u$\wedge$,iu,ie,io,ia	X					17
D,G,T,V,r(,ai,au,ei,eu,oi		X				
a+,e+,i+,o+,u+		X				15
palatal c, palatal d			X			2
ix, soft				X		2
?,Nq,V[,A:,e:,i:,o:,4:					X	8
Monolingual $\sum = 170$	40	40	30	29	31	
Multilingual						78

Outline

1. Review: Neural Network
2. Neural network based large vocabulary continuous speech recognition
3. Speech sounds: phones and phonemes
4. Bayesian vs. Discriminative Classifiers
5. How to compute the likelihood if the softmax is a posterior
6. Where we're going next

Neural net phone models: Basic setup

- Start with mel-filterbank or gammatone features, say, 40 dimensions per frame. Concatenate 11 frames together, centered at frame t, and call that $\vec{x}$ (440 dimensions).
- Define q to be the "state" at time t.
  - For now we'll say "state"="phone," $q \in \{1, \ldots, 39\}$, because there are 39 phonemes in English.
  - In a real experimental system, we might subdivide each phone into two or three states, each of which has forty or fifty context-dependent variants, which would give $q \in \mathcal{Q}$ where $|\mathcal{Q}| = 39 \times 3 \times 50 = 5850$ or so.
- The neural net output is a 39-vector $\hat{y}_t$ such that

$$\hat{y}[i] = p\left(q = i|\vec{x}\right), \quad 1 \leq i \leq 39$$

Discriminative Phone Classifier

A discriminative phone classifier chooses, in each frame,

$$q^* = \operatorname{argmax} \hat{y}[q] = \operatorname{argmax} p\left(q|\vec{x}\right)$$

- That's the optimal thing to do if $\vec{x}$ is the only information we have.
- If we have other information, then $q^* = \operatorname{argmax} \hat{y}[q]$ is suboptimal because it ignores our other information.

Other Sources of Information

What other sources of information might we have? Here are two possibilities:

1. **Information about Genre:** Maybe the test speech and training speech are about different subjects. In the training speech, a particular phoneme (say, $q = k$) never occurs, therefore the neural network always gives it a very low probability ($q(q = k|\vec{x}) \approx 0$), regardless of $\vec{x}$. We know the test genre, so we know that $q = k$ should be much more frequent. How can we fix that?
2. **Information about Sequence:** Maybe we have a language model that tells us $p(q|C)$, which is the probability of observing phone q after a particular context of preceding phones, ($C = (q_1, \ldots, q_{t-1})$). How can we combine $p(q|C)$ with $p\left(q|\vec{x}\right)$?

Bayesian Phone Classifier

A Bayesian phone classifier fuses information from multiple sources using Bayes' rule:

$$p(q=i|C,\vec{x}) = \frac{p(q=i|C)p(\vec{x}|q=i)}{\sum_j p(q=j|C)p(\vec{x}|q=j)}$$

Then, **after** performing information fusion, we can choose the most probable phone:

$$q^* = \text{argmax}\, p(q=i|C,\vec{x})$$

Now we have just one problem. How can we compute $p(\vec{x}|q)$?

The four Bayesian probabilities

A Bayesian classifier is defined if we know any row or column from the following table:

Probability Mass Functions (pmf) (must be non-negative and sum up to 1)	Probability Density Functions (pdf) (must be non-negative, but need not be less than 1)
Prior: $p(q)$	**Likelihood:** $p(\vec{x}\|q)$
Posterior: $p(q\|\vec{x})$	**Evidence:** $p(\vec{x})$

Outline

1. Review: Neural Network
2. Neural network based large vocabulary continuous speech recognition
3. Speech sounds: phones and phonemes
4. Bayesian vs. Discriminative Classifiers
5. How to compute the likelihood if the softmax is a posterior
6. Where we're going next

Why can we interpret $\hat{y}[i]$ as a posterior?

Remember why we can say $\hat{y}[i] = p\left(q = i|\vec{x}\right)$. It's because of two things:

1. $\hat{y}$ is trained using MSE for linear outputs (or using cross-entropy for softmax outputs, which has the same gradient), so, given enough training data, it learns

$$\hat{y}[i] = E\left[y[i]|\vec{x}\right] = p\left(y[i] = 1|\vec{x}\right)$$

2. $\hat{y}$ is computed using a softmax nonlinearity, which guarantees that $\hat{y}[i] > 0$ and

$$\sum_{j=1}^{39} \hat{y}[i] = 1$$

The form of the softmax nonlinearity

Remember that we define the softmax as a nonlinear transform from some excitation, $e_j^{(2)}$, to its output activation, $\hat{y}[j]$. Let's drop the superscript, so we can write

$$\hat{y}[i] = \frac{\exp(e[i])}{\sum_{j=1}^{39} \exp(e[j])}.$$

Bayes' rule

Bayes' rule defines a method for computing $p(q|\vec{x})$ in terms of $p(\vec{x}|q)$. It is

$$p(q=i|\vec{x}) = \frac{p(q=i,\vec{x})}{\sum_{j=1}^{39} p(q=j,\vec{x})}$$

Notice the similarity in those two equations. Can we make use of that?

Relationship between softmax and Bayes' rule

Suppose the neural net has been trained so that $\hat{y}[i] = p(q = i|\vec{x})$. Then we can write

$$\frac{p(q = i, \vec{x})}{\sum_{j=1}^{39} p(q = j, \vec{x})} = \frac{G \exp(e[i])}{G \sum_{j=1}^{39} \exp(e[j])}.$$

The only way this can be true is if

$$p(q = i, \vec{x}) = G \exp(e[i])$$

for some value of G. The only problem: we have no idea what G is. We have to be a little careful in our derivations, but usually we can just choose some value with good numerical properties, like $G = 1/\max_j \exp(e[j])$ for example.

Bayesian probabilities and Neural nets

Here's how we can estimate the four Bayesian probabilities using a neural net:

1. **Prior:**

$$p(q=i) = \frac{\#\text{ times } q=i \text{ occurred in training data}}{\#\text{ frames in training data}}$$

2. **Posterior:**

$$p(q=i|\vec{x}) = \text{softmax}\left(e[i]\right)$$

Bayesian probabilities and Neural nets

Here's how we can estimate the four Bayesian probabilities using a neural net:

1. **Evidence:**
$$p(\vec{x}) = G\sum_j \exp(e[j])$$
for some unknown value of G.
2. **Likelihood:**
$$p(\vec{x}|q=i) = \frac{G\exp(e[i])}{p(q=i)}$$

We have to be a little careful in our derivations, but usually we can just choose some value of G with good numerical properties, like $G = 1/\max_j \exp(e[j])$ for example.

Why the likelihood is useful

1. **Train-test mismatch**. Suppose that some particular phone ($q = i$, say) was almost never seen in training data, but it might occur sometimes in test data.
   - $p(q = i|\vec{x}) \approx 0$, because it was never seen in training data. If you classify using $q^* = \text{argmax}\, p(q|\vec{x})$, it will never get recognized.
   - $p(q = i) \approx 0$ is also very small. Therefore

     $$p(\vec{x}|q = i) = \frac{G \exp(e[i])}{p(q = i)}$$

     might be reasonably-sized, and might sometimes get recognized.
2. **Information fusion**. Suppose you have a language model that tells you $p(q|C)$, the probability of q given some context variable C. Then

   $$p(q, \vec{x}|C) = p(\vec{x}|q)p(q|C)$$

Outline

Summary: Neural Nets and Probability

The two most important Bayesian probabilities are

- **Posterior:** This is what you want if there is no train-test mismatch, and if you don't need to do information fusion. It is given by

$$p(q=i|\vec{x}) = \text{softmax}(e[i]) = \frac{\exp(e[i])}{\sum_j \exp(e[j])}$$

- **Likelihood:** This is what you want if there is train-test mismatch, or if you need to fuse information from two or more different sources. It is given by

$$p(\vec{x}|q=i) = \frac{G\exp(e[i])}{p(q=i)}$$

  where G is an unknown constant. We have to be a little careful in our derivations, but usually we can just choose some value of G with good numerical properties, like $G = 1/\max_j \exp(e[j])$ for example.

Where We're Going Next: Information Fusion

Next time, we will talk about a particular type of context information: the pronunciation model and language model, in the form of a hidden Markov model.