NLP Review

Lara J. Martin (she/they)
TA: Omkar Kulkarni (he)

https://laramartin.net/NLP -class/

Instructions for Today's Review

1. Get in groups of about 4 people
2. Go through each person's questions and teach each other
3. Make a note of any questions that couldn't be answered, things you still need clarity on, or new questions that came up
4. We'll discuss those questions as a class in the second half of class time

Classification Evaluation: Accuracy, Precision, and Recall

Min: 0 😟
Max: 1 😃

Accuracy: % of items correct

$$\frac{TP + TN}{TP + FP + FN + TN}$$

Precision: % of selected items that are correct

$$\frac{TP}{TP + FP}$$

"**Recall** measures the percentage of items actually present in the input that were correctly identified by the system." SLP, ch. 4

Recall: % of correct items that are selected

$$\frac{TP}{TP + \mathbf{FN}}$$

	Actually Target	Actually Not Target
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

F1

$$F_1 = \frac{2 * P * R}{P + R} = \frac{2 * TP}{2 * TP + FP + FN}$$

P/R/F in a Multi-class Setting: Micro- vs. Macro-Averaging

Macroaveraging: Compute performance for each class, then average.

$$\text{macroprecision} = \frac{1}{C}\sum_{c}\frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c} = \frac{1}{C}\sum_{c}\text{precision}_c$$

$$\text{macrorecall} = \frac{1}{C}\sum_{c}\frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c} = \frac{1}{C}\sum_{c}\text{recall}_c$$

Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

$$\text{microprecision} = \frac{\sum_c \text{TP}_c}{\sum_c \text{TP}_c + \sum_c \text{FP}_c}$$

$$\text{microrecall} = \frac{\sum_c \text{TP}_c}{\sum_c \text{TP}_c + \sum_c \text{FN}_c}$$

Softmax/MaxEnt

$$p(y \mid x) = \frac{\exp(\theta_y^T f(x))}{\sum_{y'} \exp(\theta_{y'}^T f(x))}$$

Maximize Log-Likelihood

$$\log \prod_i p_\theta(y_i|x_i) = \sum_i \log p_\theta(y_i|x_i)$$

Inverse of exp
log(exp(x)) = x

$$= \sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i)$$

Original maxent equation

$$\frac{\exp(\theta_y^T f(x))}{\sum_{y'} \exp(\theta_{y'}^T f(x))}$$

Differentiating this becomes nicer (even though Z depends on θ)

Minimize Cross Entropy Loss

Cross entropy:
How much $\hat{y}$ differs from the true y

Model output

True probability (i.e., correct output)

$$L^{\text{xent}}(\hat{\vec{y}}, \vec{y}) = -\sum_{k=1}^{K} \vec{y}[k] * \log p(y = k|x)$$

Probability distribution from model

index of "1" indicates correct value

$$\begin{pmatrix} 0 \\ 0 \\ \dots \\ 1 \\ \dots \\ 0 \end{pmatrix}$$

one-hot vector

SLP 6.6.1

Cosine Similarity

$$\cos(x, y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

	Dim. 1	Dim. 2	Dim. 3
apricot	2	0	0
digital	0	1	2
information	1	6	1

cosine(apricot,information) = $\frac{2+0+0}{\sqrt{4+0+0}\sqrt{1+36+1}} = 0.1622$

cosine(digital,information) = $\frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = 0.5804$

cosine(apricot,digital) = $\frac{0+0+0}{\sqrt{4+0+0}\sqrt{0+1+4}} = 0.0$

Bayes' Rule → Naïve Bayes Assumption

Bayes $$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c) \cdot P(c)}{P(d)}$$

Naïve Bayes $$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, P(c|d) \approx \underset{c \in C}{\operatorname{argmax}}\, P(d|c) \cdot P(c)$$

We can make this assumption because P(d) stays the same regardless of the class!

Perplexity

Lower is better : lower perplexity ➔ less surprised

$$\text{perplexity} = \exp\left(\frac{-1}{M} \sum_{i=1}^{M} \log p(w_i \mid h_i)\right)$$

$$= \sqrt[M]{\prod_{i=1} \frac{1}{p(w_i \mid h_i)}}$$

weighted geometric average

Trigram MLE

p(Colorless green ideas sleep furiously) =
p(Colorless | *<BOS>* *<BOS>*) *
p(green | *<BOS>* Colorless) *
p(ideas | Colorless green) *
p(sleep | green ideas) *
p(furiously | ideas sleep) *
p(*<EOS>* | sleep furiously)

Consistent notation: Pad the left with <BOS> (beginning of sentence) symbols
Fully proper distribution: Pad the right with a single <EOS> symbol

Add-λ estimation

Other names: Laplace smoothing, Lidstone smoothing

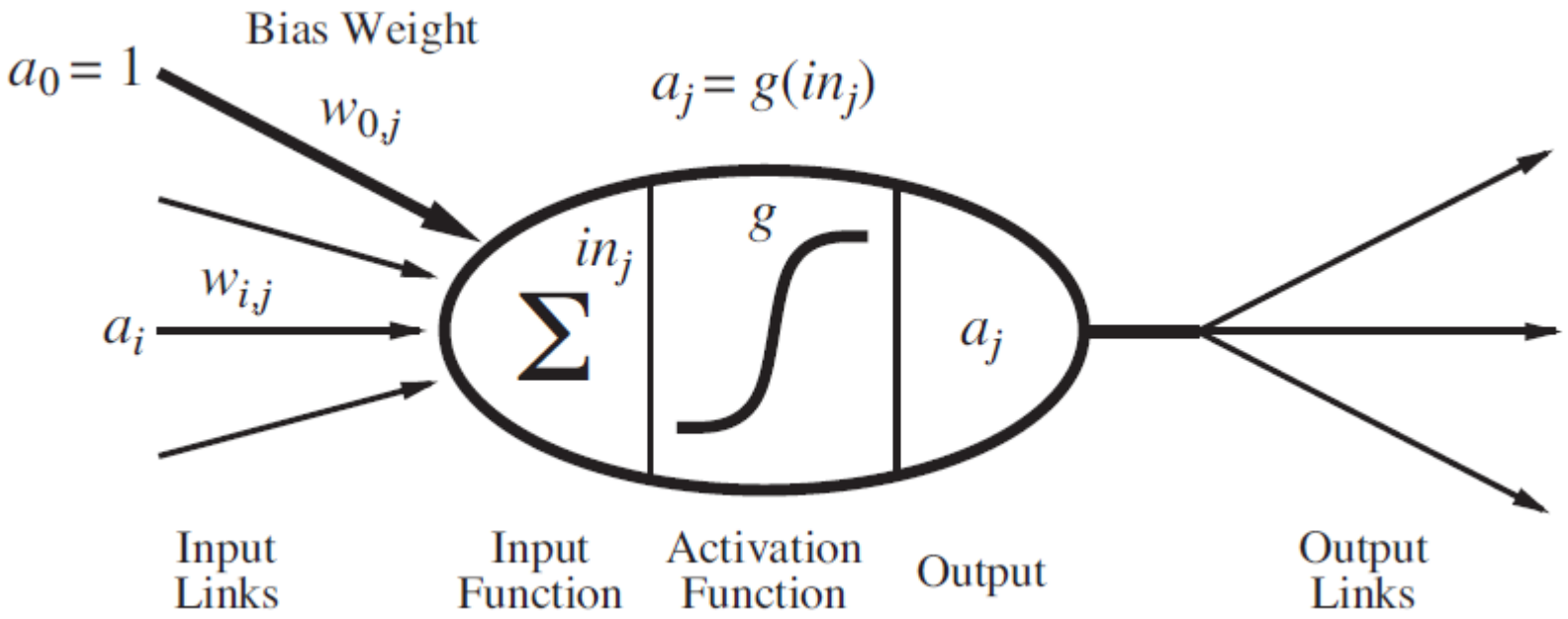
Pretend we saw each word λ more times than we did

Add λ to all the counts

$$p(\mathrm{z}) \propto count(\mathrm{z}) + \lambda$$
$$= \frac{count(\mathrm{z}) + \lambda}{\sum_v (count(\mathrm{v}) + \lambda)}$$

Biologically-Inspired Learning Models: Neuron Unit

$$in_j = w_{0j} + w_{1j}a_1 + w_{2j}a_2 + \cdots + w_{ij}a_i$$



activations

$0 \leq a_i \leq 1$

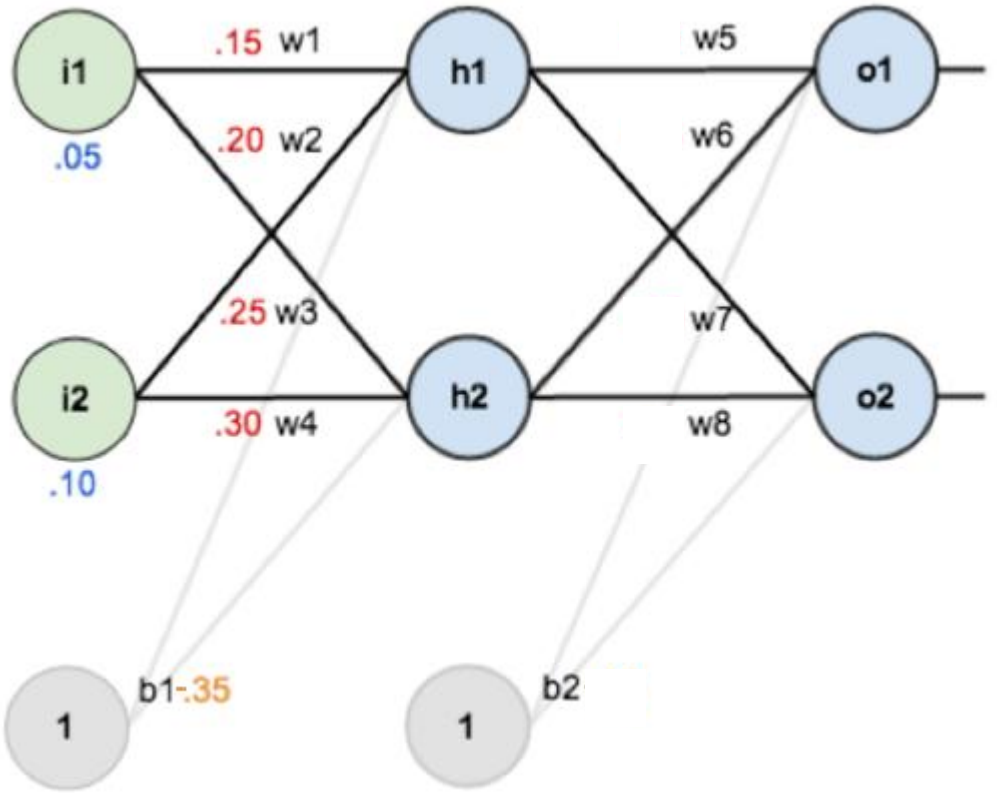
weights

$-\infty < w_{ij} < \infty$

Forward Propagation Example

(1) $in_j = w_{0j} + w_{1j}a_1 + w_{2j}a_2 + \cdots + w_{ij}a_i$

in_{h1} =–.3225
in_{h2} = –.3075



For each layer:
1. Calculate the weighted sum of inputs to each neuron unit
2. Evaluate the activation function to determine the output of each neuron unit
3. Use outputs as inputs for the next layer

(2) Activation function

Sigmoid

$$g(x) = \frac{1}{1+e^{-x}}$$

$out_{h1} = g(in_{h1})$
$= \frac{1}{1+e^{-in_{h1}}}$
$= \frac{1}{1+e^{-(-.3275)}}$
= .4188

$out_{h2} = g(in_{h2})$
$= \frac{1}{1+e^{-in_{h2}}}$
$= \frac{1}{1+e^{-(-.3075)}}$
= .4237