**N-grams**

N-gram-based language models are often used to compute the probability of a sentence \( W \):

\[
P(W) = \prod_i (w_i|w_1 \ldots w_{i-1})
\]

Often simplified to trigrams:

\[
P(W) = \prod_i (w_i|w_{i-2}, w_{i-1})
\]

\[
P\text{(this is a sentence)} = P\text{(this)} \times P\text{(is|this)} \times P\text{(a|this, is)} \times P\text{(sentence|is, a)}
\]

\[
P\text{(a|this, is)} = \frac{C\text{(this is a)}}{C\text{(this is)}}
\]

**One-hot Representations**

Simple way how to encode discrete concepts, such as words; also known as 1-of-\( N \) where \( N \) would be the size of the vocabulary.

<table>
<thead>
<tr>
<th>vocabulary = (Monday, Tuesday, is, a, today)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday = [1 0 0 0 0]</td>
</tr>
<tr>
<td>Tuesday = [0 1 0 0 0]</td>
</tr>
<tr>
<td>is = [0 0 1 0 0]</td>
</tr>
<tr>
<td>a = [0 0 0 1 0]</td>
</tr>
<tr>
<td>today = [0 0 0 0 1]</td>
</tr>
</tbody>
</table>

**Bag-of-Words Representations**

Ignores word order, sum of one-hot vectors. Can be extended to bag-of-N-grams to capture local ordering of words.

<table>
<thead>
<tr>
<th>vocabulary = (Monday, Tuesday, is, a, today)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday Monday = [2 0 0 0 0]</td>
</tr>
<tr>
<td>today is a Monday = [1 0 1 1 1]</td>
</tr>
<tr>
<td>today is a Tuesday = [0 1 1 1 1]</td>
</tr>
<tr>
<td>is a Monday today = [1 0 1 1 1]</td>
</tr>
</tbody>
</table>
Logistic Regression

- **Input** is a feature vector, **output** is one (binary classification) or many (multinomial distribution).
- Weights matrix (or vector) **directly** connects inputs and output.
- Trained by gradient descent (neural network; no hidden units).

```
\begin{align*}
  &x_1 \\
  &x_2 \\
  &x_3 \\
  &x_4 \\
  &x_5 \\
  \end{align*}
\begin{align*}
  &w_1 \\
  &w_2 \\
  &w_3 \\
  &w_4 \\
  &w_5 \\
  \end{align*}
```

Neural Networks

- **Activation function**: sigmoid, tanh

```
W: input weights, matrix
I: input signal, feature vector (one per example)
```

Training of Neural Networks

**Forward Pass**
- Input signal is presented first.
- Hidden layer state is computed (vector times matrix operation and non-linear activation).
- Outputs are computed (vectors times matrix operation and usually non-linear activation).

**Backpropagation**
- To train the network, we need to compute gradient of the error.
- The gradients are sent back using the same weights that were used in the forward pass.

**Learning Rate**:
- Controls how much we change the weights.
- Too little value will result in long training time, too high value will erase previously learned patterns.
- We start with high learning rate and reduce it during training.

**Training Epochs**:
- Several passes over the training data are often performed (epoch: number of iterations over the data set in order to train the neural network).

**Regularization**:
- Network often overfits (it fails to generalize at test time).
- High weights are used to model only some small subset of data.
- We can try to force the weights to stay small during training to avoid this problem (L1 & L2 regularization).
Model Parameters

Choice of the **hyper-parameters** has to be done manually:
- Type of activation function
- Choice of architecture (how many hidden layers, their sizes)
- Learning rate, number of training epochs
- What features are presented at the input layer
- How to regularize

---

A Basic Neural Language Model

- Bigram neural language model
- Previous word predicts current word via hidden layer
- As many outputs as there are words in the vocabulary
- Model learns compressed, continuous representations of words (usually the matrix of weights between input and hidden layer)

---

Another Neural Language Model

Bengio et al. (2003, JMLR)

- Input layer, projection layer, hidden layer and output layer
- The projection layer is linear
- The hidden layer is non-linear
- Softmax at the output computes probability distribution over the whole vocabulary ($g(o) = \frac{e^{o_i}}{\sum_{k} e^{o_k}}$)
- Model is computationally very expensive
Continuous bag-of-words model (CBOW)

- Mikolov et al. (2013, ICLR)
- CBOW adds inputs from words within short window to predict the current word
- The weights for different positions are shared
- Computationally much more efficient than normal NNLM
- The hidden layer is just linear

Skip-gram NNLM

- We can reformulate the CBOW model by predicting surrounding words using the current word
- If both CBOW and skip-gram NNLM are trained for sufficient number of epochs, their performance is similar

Training

- Stochastic gradient descent and backpropagation
- It is useful to sub-sample the frequent words (e.g., the, is, a)
- Words are thrown out proportional to their frequency (makes things faster, reduces importance of frequent words like IDF)
- Non-linearity does not seem to improve performance of these models, thus the hidden layer does not use activation function
- Problem: very large output layer - size equal to vocabulary size, can easily be in order of millions (too many outputs to evaluate)
- Solution: negative sampling (also Hierarchical softmax)

Negative Sampling

- Instead of propagating signal from the hidden layer to the whole output layer, only the output neuron that represents the positive class + few randomly sampled neurons are evaluated
- The output neurons are treated as independent logistic regression classifiers
- This makes the training speed independent of the vocabulary size (can be easily parallelized)
- 20K questions, 6B words, 1M words vocabulary
- Find word closest to question word (e.g., Greece), consider it correct if it matches the answer.

Can you think of additional baseline comparisons?

Subtracting two word vectors, and add the result to another word.
Summary

Don’t count predict! Baroni et al. (ACL, 2014)

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word2vec</td>
<td>Available at <a href="https://code.google.com/p/word2vec/">https://code.google.com/p/word2vec/</a></td>
</tr>
<tr>
<td></td>
<td>Tool for training the word vectors using CBOW and skip-gram architectures, supports both negative sampling and hierarchical softmax</td>
</tr>
<tr>
<td></td>
<td>Optimized for very large datasets (&gt;billions of training words)</td>
</tr>
<tr>
<td></td>
<td>Pre-trained on large datasets (100B words)</td>
</tr>
<tr>
<td></td>
<td>Continuous Space Language Model toolkit: <a href="http://www-lium.univ-lemans.fr/cslm/">http://www-lium.univ-lemans.fr/cslm/</a></td>
</tr>
<tr>
<td></td>
<td>Feedforward neural network language model (Holger Schwenk)</td>
</tr>
</tbody>
</table>