CARTE ML Workshop

Embeddings

Outline

Embeddings

Outline

Embeddings

From Real to Symbolic

- Previously, we have looked at models that deal with real-valued inputs
- This means that the input is already a number, or can be easily converted to a number
- But what if the input is a symbol?

Symbolic variable

- Text: characters, words, bigrams...
- Recommender Systems: item ids, user ids
- Any categorical descriptor: tags, movie genres, visited URLs, skills on a resume, product categories...

Symbolic variable

- Text: characters, words, bigrams...
- Recommender Systems: item ids, user ids
- Any categorical descriptor: tags, movie genres, visited URLs, skills on a resume, product categories...

Symbolic variable

- Text: characters, words, bigrams...
- Recommender Systems: item ids, user ids
- Any categorical descriptor: tags, movie genres, visited URLs, skills on a resume, product categories...

Symbolic variable

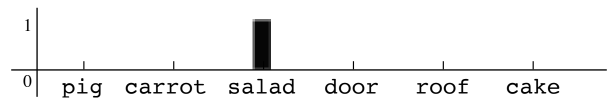
- Text: characters, words, bigrams...
- Recommender Systems: item ids, user ids
- Any categorical descriptor: tags, movie genres, visited URLs, skills on a resume, product categories...

Notation:

$$\text{Symbol } s \text{ in vocabulary } V$$

One-hot representation

$$\textit{onehot}(\text{'salad'}) = [0, 0, 1, \ldots, 0] \in \{0, 1\}^{|V|}$$



One-hot representation

$$onehot(\text{'salad'}) = [0, 0, 1, \ldots, 0] \in \{0, 1\}^{|V|}$$

- Sparse, discrete, large dimension $|V|$
- Each axis has a meaning
- Symbols are equidistant from each other:

euclidean distance = $\sqrt{2}$

Embedding

$$embedding(\text{'salad'}) = [3.28, -0.45, \ldots 7.11]$$

Embedding

$$embedding(\text{'salad'}) = [3.28, -0.45, \ldots 7.11]$$

- Continuous and dense
- Can represent a huge vocabulary in low dimension, typically: $d \in \{16, 32, \ldots, 4096\}$
- Axis have no meaning *a priori*
- Embedding metric can capture semantic distance

Embedding

$$embedding(\text{'salad'}) = [3.28, -0.45, \ldots 7.11]$$

- Continuous and dense
- Can represent a huge vocabulary in low dimension, typically: $d \in \{16, 32, \ldots, 4096\}$
- Axis have no meaning *a priori*
- Embedding metric can capture semantic distance

Neural Networks compute transformations on continuous vectors

Implementation with Keras

Size of vocabulary $n = |V|$, size of embedding d

```
# input: batch of integers
Embedding(output_dim=d, input_dim=n, input_length=1)
# output: batch of float vectors
```

Implementation with Keras

Size of vocabulary $n = |V|$, size of embedding d

```
# input: batch of integers
Embedding(output_dim=d, input_dim=n, input_length=1)
# output: batch of float vectors
```

- Equivalent to one-hot encoding multiplied by a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$:

$$embedding(x) = onehot(x). \mathbf{W}$$

Implementation with Keras

Size of vocabulary $n = |V|$, size of embedding d

```
# input: batch of integers
Embedding(output_dim=d, input_dim=n, input_length=1)
# output: batch of float vectors
```

- Equivalent to one-hot encoding multiplied by a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$:

$$embedding(x) = onehot(x). \mathbf{W}$$

- $\mathbf{W}$ is typically **randomly initialized**, then **tuned by backprop**

Implementation with Keras

Size of vocabulary $n = |V|$, size of embedding d

```
# input: batch of integers
Embedding(output_dim=d, input_dim=n, input_length=1)
# output: batch of float vectors
```

- Equivalent to one-hot encoding multiplied by a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$:

$$embedding(x) = onehot(x). \mathbf{W}$$

- $\mathbf{W}$ is typically **randomly initialized**, then **tuned by backprop**
- $\mathbf{W}$ are trainable parameters of the model

Distance and similarity in Embedding space

Euclidean distance

$d(x, y) = ||x - y||_2$

- Simple with good properties
- Dependent on norm (embeddings usually unconstrained)

Distance and similarity in Embedding space

Euclidean distance

$d(x, y) = ||x - y||_2$

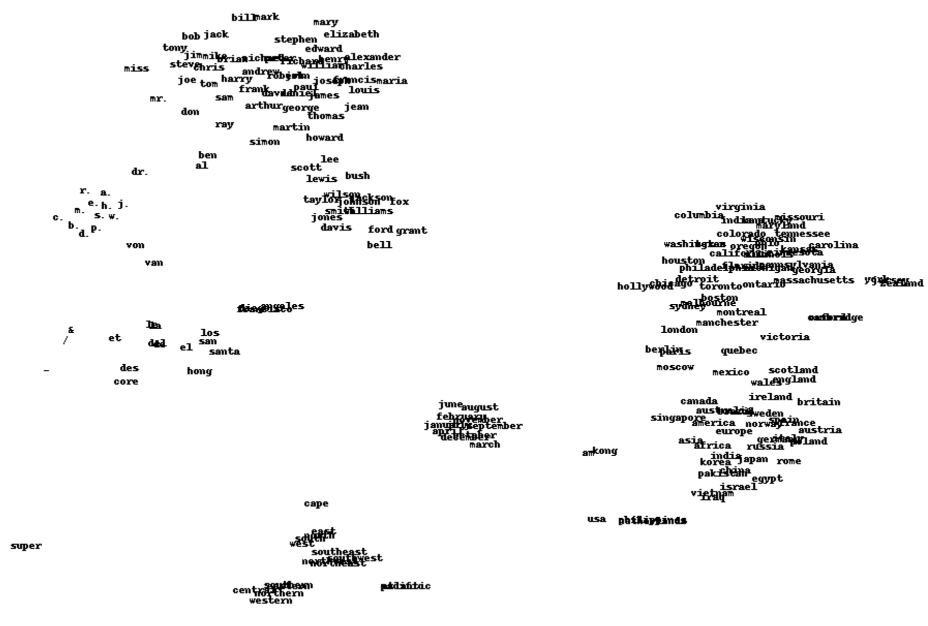
- Simple with good properties
- Dependent on norm (embeddings usually unconstrained)

Cosine similarity

$cosine(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||}$

- Angle between points, regardless of norm
- $cosine(x, y) \in (-1, 1)$
- Expected cosine similarity of random pairs of vectors is 0

Example word vectors



excerpt from work by J. Turian on a model trained by R. Collobert et al. 2008

Dropout Regularization

Regularization

Width of the network

Regularization

Width of the network

Depth of the network

Regularization

Width of the network

Depth of the network

L_2 penalty on weights

Regularization

Width of the network

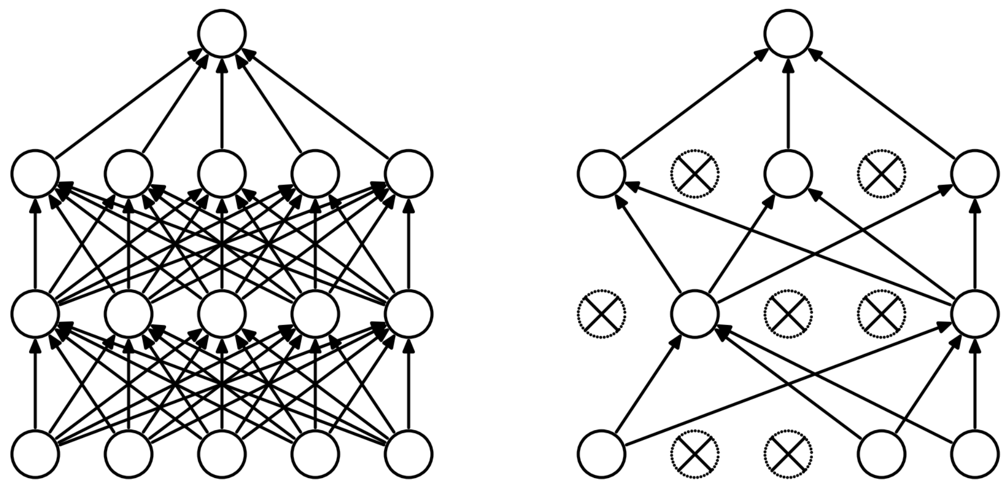
Depth of the network

L_2 penalty on weights

Dropout

- Randomly set activations to 0 with probability p
- Typically only enabled at training time

Dropout



(a) Standard Neural Net (b) After applying dropout.

Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Srivastava et al., *Journal of Machine Learning Research* 2014

Dropout

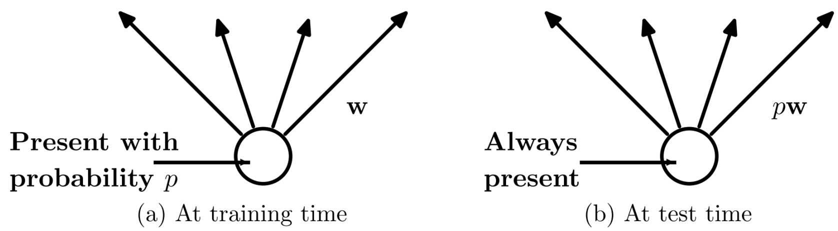
Interpretation

- Reduces the network dependency to individual neurons
- More redundant representation of data

Ensemble interpretation

- Equivalent to training a large ensemble of shared-parameters, binary-masked models
- Each model is only trained on a single data point

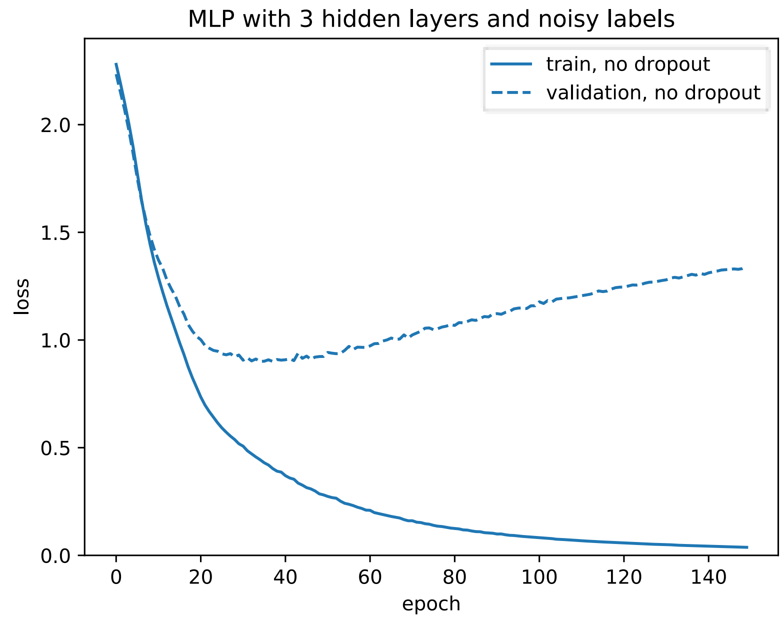
Dropout



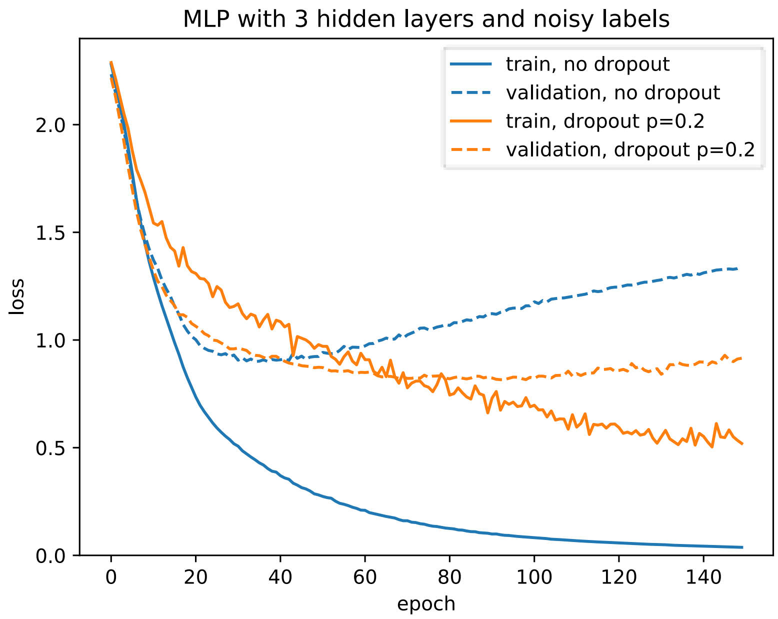
At test time, multiply weights by p to keep same level of activation

Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Srivastava et al., *Journal of Machine Learning Research* 2014

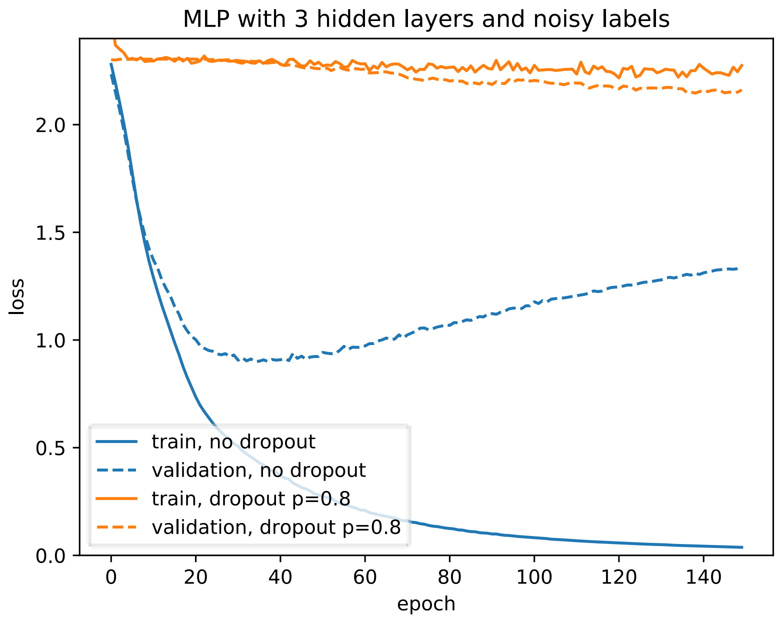
Overfitting Noise



A bit of Dropout



Too much: Underfitting



Implementation with Keras

```
model = Sequential()
model.add(Dense(hidden_size, input_shape, activation='relu'))
model.add(Dropout(p=0.5))
model.add(Dense(hidden_size, activation='relu'))
model.add(Dropout(p=0.5))
model.add(Dense(output_size, activation='softmax'))
```