Feed-Forward Neural Network Design

Yiming Mao

1 Linear Regression

Encoding for the Inputs

Each predictor is a real-valued scalar feature such as a measurement or numerical variable. The input vector is constructed as

$$\mathbf{x} = [x_1, x_2, \ldots, x_d]^\top \in \mathbb{R}^d.$$

Since the inputs are already real numbers, no embedding is required. Optionally, each feature may be standardized to zero mean and unit variance for numerical stability.

Sequence of Layers

The model uses a single fully connected layer:

$$\hat{y} = \mathbf{W}\mathbf{x} + \mathbf{b},$$

where

$$\mathbf{W} \in \mathbb{R}^{1\times d}, \quad \mathbf{b} \in \mathbb{R}.$$

No activation function is used (purely linear mapping). A bias term is included to allow the regression line to shift vertically.

Loss Function

Mean Squared Error (MSE):

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2.$$

2 Classification: NFL Player Position

Encoding for the Inputs

The two predictors, height and weight, are real-valued continuous variables. They are represented as a 2D vector:

$$\mathbf{x} = [h, w]^\top \in \mathbb{R}^2.$$

To ensure balanced scaling, both height and weight can be normalized to zero mean and unit variance.

Sequence of Layers

$$\begin{aligned} \mathbf{h} &= \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1), & \mathbf{W}_1 &\in \mathbb{R}^{8\times 2},\ \mathbf{b}_1 \in \mathbb{R}^8, \\ \hat{\mathbf{y}} &= \text{Softmax}(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2), & \mathbf{W}_2 &\in \mathbb{R}^{3\times 8},\ \mathbf{b}_2 \in \mathbb{R}^3. \end{aligned}$$

Design choices:

- Hidden layer uses ReLU activation to introduce nonlinearity.
- Bias vectors are included to improve model flexibility.
- Output layer uses Softmax to produce class probabilities over the three positions.

Loss Function

Categorical Cross-Entropy:

$$\mathcal{L} = -\sum_{k=1}^{3} y_k \log \hat{y}_k.$$

3 Bigram Language Modeling

Encoding for the Inputs

Each token t_i is represented as a one-hot vector

$$\mathbf{x}_i \in \{0,1\}^V,$$

where V is the vocabulary size. To convert this sparse representation into a dense, fixed-length real vector, we use an embedding matrix:

$$\mathbf{e}_i = E\mathbf{x}_i, \quad E \in \mathbb{R}^{d \times V}.$$

The resulting vector $\mathbf{e}_i \in \mathbb{R}^d$ encodes semantic information about the token.

Sequence of Layers

A single linear transformation projects the embedding to vocabulary logits:

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{W}\mathbf{e}_i + \mathbf{b}),$$

where

$$\mathbf{W} \in \mathbb{R}^{V \times d}, \quad \mathbf{b} \in \mathbb{R}^V.$$

Design choices:

- The embedding layer fixes input size at d.
- A bias vector is added to improve modeling flexibility.
- Softmax activation converts raw scores into a probability distribution over vocabulary tokens.

Loss Function

Cross-Entropy Loss:

$$\mathcal{L} = -\sum_{v=1}^{V} y_v \log \hat{y}_v.$$

4 Trigram Language Modeling

Encoding for the Inputs

Each of the two preceding tokens, t_{i-1} and t_i, is represented as a one-hot vector and mapped to embeddings:

$$\mathbf{e}_{i-1} = E\mathbf{x}_{i-1}, \quad \mathbf{e}_i = E\mathbf{x}_i,$$

with $E \in \mathbb{R}^{d \times V}$. We concatenate the two embeddings to form a fixed-size input vector:

$$\mathbf{x} = [\mathbf{e}_{i-1}; \mathbf{e}_i] \in \mathbb{R}^{2d}.$$

Sequence of Layers

A single fully connected layer predicts the next-token distribution:

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

where

$$\mathbf{W} \in \mathbb{R}^{V \times 2d}, \quad \mathbf{b} \in \mathbb{R}^V.$$

Design choices:

- Input size is $2d$ because two embeddings are concatenated.
- Bias vector is used.
- Softmax activation ensures probabilities sum to one.

Loss Function

Cross-Entropy Loss:

$$\mathcal{L} = -\sum_{v=1}^{V} y_v \log \hat{y}_v.$$

5 Trigram Language Modeling (Improved)

Encoding for the Inputs

Same as before — the two context tokens are embedded and concatenated:

$$\mathbf{x} = [\mathbf{e}_{i-1}; \mathbf{e}_i] \in \mathbb{R}^{2d}.$$

This ensures a fixed-size, dense input representation that captures both word identities.

Sequence of Layers

To increase expressiveness and reduce underfitting, we add two nonlinear hidden layers:

$$\begin{aligned}
\mathbf{h}_1 &= \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1), \quad \mathbf{W}_1 \in \mathbb{R}^{256\times 2d},\ \mathbf{b}_1 \in \mathbb{R}^{256}, \\
\mathbf{h}_2 &= \text{ReLU}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2), \quad \mathbf{W}_2 \in \mathbb{R}^{128\times 256},\ \mathbf{b}_2 \in \mathbb{R}^{128}, \\
\hat{\mathbf{y}} &= \text{Softmax}(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3), \quad \mathbf{W}_3 \in \mathbb{R}^{V\times 128},\ \mathbf{b}_3 \in \mathbb{R}^{V}.
\end{aligned}$$

Design choices:

- Two hidden layers with ReLU activations capture nonlinear dependencies.
- Biases are included in all layers to enhance flexibility.
- Output Softmax ensures normalized probabilities.

Loss Function

Cross-Entropy Loss:

$$\mathcal{L} = -\sum_{v=1}^{V} y_v \log \hat{y}_v.$$

Remarks

The deeper architecture enables the model to represent complex syntactic and semantic relationships in the context. Hidden dimensions (256 and 128) provide sufficient capacity to fit the data without excessive overfitting. Dropout or normalization can be introduced for regularization.