# Introduction to Deep Learning for Speech and Text Processing
# Exercise Sheet 4: Machine Learning - Basics

Thang Vu

7th November 2025

## Clustering

**Exercise 1.**
In speaker recognition or speech synthesis the information from speakers is often represented in high dimensional *speaker embedding vectors* $\in \mathbb{R}^n$. The *speaker embedding vectors* are computed on utterance level, i.e. we compute a single vector for each utterance. Now, given a set of utterances from two different speakers, we want to evaluate how well our embedding function performs.

The reduced speaker embedding vectors on utterance level are as follows:

$$v(0) = \begin{bmatrix}1\\1\end{bmatrix}, v(1) = \begin{bmatrix}1\\1.5\end{bmatrix}, v(2) = \begin{bmatrix}1.5\\1.5\end{bmatrix}, v(3) = \begin{bmatrix}2\\2\end{bmatrix}, v(4) = \begin{bmatrix}2.5\\3\end{bmatrix}, v(5) = \begin{bmatrix}2\\1\end{bmatrix}, v(6) = \begin{bmatrix}2.5\\1\end{bmatrix}, v(7) = \begin{bmatrix}3\\1\end{bmatrix}, v(8) = \begin{bmatrix}2.5\\1.5\end{bmatrix}$$

Perform two update steps using the k-means algorithm with $k = 2$ clusters. The initial means are $v(1)$ and $v(7)$. Use euclidean distance for this task and assign every data point to its closest centroid.

**Solution 1.**
**Iteration 1.**

**Initial centroids:**

$$c_A^0 = (1, 1.5), \qquad c_B^0 = (3, 1).$$

**Distance calculations (Euclidean).**

For $v(0) = (1, 1)$:

$$d(v(0), c_A^0) = \sqrt{(1-1)^2 + (1-1.5)^2} = \sqrt{0 + 0.25} = 0.5, \qquad d(v(0), c_B^0) = \sqrt{(1-3)^2 + (1-1)^2} = \sqrt{4+0} = 2,$$

thus $v(0) \to A$.

For $v(1) = (1, 1.5)$: $d(v(1), c_A^0) = 0$, $d(v(1), c_B^0) = \sqrt{4.25} \approx 2.062$, thus $v(1) \to A$.

For $v(2) = (1.5, 1.5)$: $d(v(2), c_A^0) = 0.5$, $d(v(2), c_B^0) = \sqrt{2.5} \approx 1.581$, thus $v(2) \to A$.

For $v(3) = (2, 2)$: $d(v(3), c_A^0) = \sqrt{1.25} \approx 1.118$, $d(v(3), c_B^0) = \sqrt{2} \approx 1.414$, thus $v(3) \to A$.

For $v(4) = (2.5, 3)$: $d(v(4), c_A^0) = \sqrt{4.5} \approx 2.121$, $d(v(4), c_B^0) = \sqrt{4.25} \approx 2.062$, thus $v(4) \to B$.

For $v(5) = (2, 1)$: $d(v(5), c_A^0) = \sqrt{1.25} \approx 1.118$, $d(v(5), c_B^0) = 1$, thus $v(5) \to B$.

For $v(6) = (2.5, 1)$: $d(v(6), c_A^0) = \sqrt{2.5} \approx 1.581$, $d(v(6), c_B^0) = 0.5$, thus $v(6) \to B$.

For $v(7) = (3, 1)$: $d(v(7), c_A^0) = \sqrt{4.25} \approx 2.062$, $d(v(7), c_B^0) = 0$, thus $v(7) \to B$.

For $v(8) = (2.5, 1.5)$: $d(v(8), c_A^0) = 1.5$, $d(v(8), c_B^0) = \sqrt{0.5} \approx 0.707$, thus $v(8) \to B$.

After iteration 1 the clusters are

$$A = \{v(0), v(1), v(2), v(3)\}, \qquad B = \{v(4), v(5), v(6), v(7), v(8)\}.$$

**Update step:**

$$c_A^1 = \frac{(1,1) + (1,1.5) + (1.5,1.5) + (2,2)}{4} = \left(\frac{11}{8}, \frac{3}{2}\right) = (1.375, 1.5),$$

$$c_B^1 = \frac{(2.5,3) + (2,1) + (2.5,1) + (3,1) + (2.5,1.5)}{5} = \left(\frac{5}{2}, \frac{3}{2}\right) = (2.5, 1.5).$$

**Iteration 2.**

For $v(0) = (1,1)$:

$$d(v(0), c_A^1) = \sqrt{(1-1.375)^2 + (1-1.5)^2} = \sqrt{(-0.375)^2 + (-0.5)^2} = \sqrt{0.140625 + 0.25} = \sqrt{0.390625} = 0.625,$$

$$d(v(0), c_B^1) = \sqrt{(1-2.5)^2 + (1-1.5)^2} = \sqrt{(-1.5)^2 + (-0.5)^2} = \sqrt{2.25 + 0.25} = \sqrt{2.5} \approx 1.581,$$

thus $v(0) \to A$.

For $v(1) = (1, 1.5)$: $d(v(1), c_A^1) = 0.375,\ d(v(1), c_B^1) = 1.5$, thus $v(1) \to A$.

For $v(2) = (1.5, 1.5)$: $d(v(2), c_A^1) = 0.125,\ d(v(2), c_B^1) = 1$, thus $v(2) \to A$.

For $v(3) = (2, 2)$: $d(v(3), c_A^1) = \sqrt{0.640625} \approx 0.800,\ d(v(3), c_B^1) = \sqrt{0.5} \approx 0.707$, thus $v(3) \to B$.

For $v(4) = (2.5, 3)$: $d(v(4), c_A^1) = 1.875,\ d(v(4), c_B^1) = 1.5$, thus $v(4) \to B$.

For $v(5) = (2, 1)$: $d(v(5), c_A^1) = \sqrt{0.640625} \approx 0.800,\ d(v(5), c_B^1) = \sqrt{0.5} \approx 0.707$, thus $v(5) \to B$.

For $v(6) = (2.5, 1)$: $d(v(6), c_A^1) = \sqrt{1.515625} \approx 1.231,\ d(v(6), c_B^1) = 0.5$, thus $v(6) \to B$.

For $v(7) = (3, 1)$: $d(v(7), c_A^1) = \sqrt{2.890625} \approx 1.700,\ d(v(7), c_B^1) = \sqrt{0.5} \approx 0.707$, thus $v(7) \to B$.

For $v(8) = (2.5, 1.5)$: $d(v(8), c_A^1) = 1.125,\ d(v(8), c_B^1) = 0$, thus $v(8) \to B$.

After iteration 2 the clusters are

$$A = \{v(0), v(1), v(2)\}, \qquad B = \{v(3), v(4), v(5), v(6), v(7), v(8)\}.$$

**Update step:**

$$c_A^2 = \frac{(1,1) + (1,1.5) + (1.5,1.5)}{3} = \left(\frac{7}{6}, \frac{4}{3}\right) \approx (1.1667,\ 1.3333),$$

$$c_B^2 = \frac{(2,2) + (2.5,3) + (2,1) + (2.5,1) + (3,1) + (2.5,1.5)}{6} = \left(\frac{29}{12}, \frac{19}{12}\right) \approx (2.4167,\ 1.5833).$$

## Features

**Exercise 2.**

The primary goal of sentiment analysis is to understand and categorize the opinions expressed in texts. It can be considered as a classification task in machine learning. In a traditional machine learning pipeline, the following features are reasonable:

(1) Word features: the average word embedding of all words of a sentence

(2) Aggregated sentiment scores: the sum of all sentiment scores of each word

(3) Last word sentiment: the sentiment score of the last word

Given the following word embeddings:

$$v('it') = \begin{bmatrix}0.5\\0.5\\0\end{bmatrix}, v('this') = \begin{bmatrix}0.4\\0.6\\-0.7\end{bmatrix}, v('is') = \begin{bmatrix}0\\0\\0.8\end{bmatrix}, v('movie') = \begin{bmatrix}0.2\\0.4\\-0.8\end{bmatrix}, v('a') = \begin{bmatrix}0.2\\-0,2\\0\end{bmatrix}$$

$$v('good') = \begin{bmatrix}0.7\\0\\0\end{bmatrix}, v('bad') = \begin{bmatrix}-0.8\\0\\0.1\end{bmatrix}, v('great') = \begin{bmatrix}0.9\\0.1\\0\end{bmatrix}, s('not') = \begin{bmatrix}-0.2\\-0.2\\0.1\end{bmatrix}$$

and the following sentiment dictionary:

$$s('it') = 0, s('this') = 0, s('is') = 0, s('movie') = 0.2, s('a') = 0$$

$$s('good') = 0.8, s('bad') = -0.6, v('great') = 0.7, s('not') = -0.2$$

- extract the aforementioned features and construct the feature vector of the following inputs:
  (1) it is a great movie
  (2) this movie is not a good movie
  (3) it is a bad movie
- which features have low variance and might not be useful based on the above examples?

**Solution 2.**

(1) "it is a great movie"

Tokens: it, is, a, great, movie ($T = 5$).

**Word features:** $\sum v = \begin{bmatrix}0.5\\0.5\\0\end{bmatrix} + \begin{bmatrix}0\\0\\0.8\end{bmatrix} + \begin{bmatrix}0.2\\-0.2\\0\end{bmatrix} + \begin{bmatrix}0.9\\0.1\\0\end{bmatrix} + \begin{bmatrix}0.2\\0.4\\-0.8\end{bmatrix} = \begin{bmatrix}1.8\\0.8\\0\end{bmatrix}$, so $\bar{v} = \frac{1}{5}\begin{bmatrix}1.8\\0.8\\0\end{bmatrix} = \begin{bmatrix}0.36\\0.16\\0\end{bmatrix}$.

**Aggregated sentiment:** $S = 0 + 0 + 0 + 0.7 + 0.2 = 0.9$.

**Last-word sentiment:** $\ell = s(\text{movie}) = 0.2$.

(2) "this movie is not a good movie"

Tokens: this, movie, is, not, a, good, movie ($T = 7$).

**Word features:** $\sum v = \begin{bmatrix}0.4\\0.6\\-0.7\end{bmatrix} + \begin{bmatrix}0.2\\0.4\\-0.8\end{bmatrix} + \begin{bmatrix}0\\0\\0.8\end{bmatrix} + \begin{bmatrix}-0.2\\-0.2\\0.1\end{bmatrix} + \begin{bmatrix}0.2\\-0.2\\0\end{bmatrix} + \begin{bmatrix}0.7\\0\\0\end{bmatrix} + \begin{bmatrix}0.2\\0.4\\-0.8\end{bmatrix} = \begin{bmatrix}1.5\\1.0\\-1.4\end{bmatrix}$, so $\bar{v} = \frac{1}{7}\begin{bmatrix}1.5\\1.0\\-1.4\end{bmatrix} = \begin{bmatrix}0.2142857\\0.1428571\\-0.2\end{bmatrix}$.

**Aggregated sentiment:** $S = 1.0$.

**Last-word sentiment:** $\ell = s(\text{movie}) = 0.2$.

(3) "it is a bad movie"

Tokens: it, is, a, bad, movie ($T = 5$).

**Word features:** $\sum v = \begin{bmatrix}0.5\\0.5\\0\end{bmatrix} + \begin{bmatrix}0\\0\\0.8\end{bmatrix} + \begin{bmatrix}0.2\\-0.2\\0\end{bmatrix} + \begin{bmatrix}-0.8\\0\\0.1\end{bmatrix} + \begin{bmatrix}0.2\\0.4\\-0.8\end{bmatrix} = \begin{bmatrix}0.1\\0.7\\0.1\end{bmatrix}$, so $\bar{v} = \frac{1}{5}\begin{bmatrix}0.1\\0.7\\0.1\end{bmatrix} = \begin{bmatrix}0.02\\0.14\\0.02\end{bmatrix}$.

**Aggregated sentiment:** $S = -0.4$.

**Last-word sentiment:** $\ell = s(\text{movie}) = 0.2$.

**Low-variance observation**

For all three sentences, the last token is *movie* with $s(\text{movie}) = 0.2$, so the last-word sentiment has zero variance and is uninformative, while word features and aggregated sentiment vary across sentences.

## Logistic Regression

**Exercise 3.**
Compute the predictions for the following logistic regression model $p(y = 1|x)$ and the following test dataset. The model is trained to binary-classify sentences: a label of 1 refers to a positive sentiment, a label of 0 to a negative sentiment.

Logistic Regression Model:

$$w = \begin{bmatrix} -2 \\ 4 \\ -1 \\ 0.5 \end{bmatrix}, \quad b = -1$$

Dataset:

$$(x_1 = \begin{bmatrix} -1 \\ 2 \\ 0 \\ -2 \end{bmatrix}, \quad y_1 = 1), \quad (x_2 = \begin{bmatrix} 2 \\ 0 \\ 2.5 \\ 2 \end{bmatrix}, \quad y_2 = 0), \quad (x_3 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 0 \end{bmatrix}, \quad y_3 = 1) \quad (x_4 = \begin{bmatrix} 1 \\ 0.5 \\ -1 \\ 0 \end{bmatrix}, \quad y_4 = 1)$$

How good is the model in your opinion? Explain your intuition.

**Solution 3.**
**Prediction for $x_1$:**

$$w^\top x_1 + b = (-2)(-1) + 4(2) + (-1)(0) + 0.5(-2) - 1 = 2 + 8 + 0 - 1 - 1 = 8.$$

$$p(y = 1|x_1) = \sigma(8) = \frac{1}{1 + e^{-8}} \approx 0.9997 \Rightarrow \hat{y}_1 = 1$$

(correct).

**Prediction for $x_2$:**

$$w^\top x_2 + b = (-2)(2) + 4(0) + (-1)(2.5) + 0.5(2) - 1 = -4 + 0 - 2.5 + 1 - 1 = -6.5.$$

$$p(y = 1|x_2) = \sigma(-6.5) = \frac{1}{1 + e^{6.5}} \approx 0.0015 \Rightarrow \hat{y}_2 = 0$$

(correct).

**Prediction for $x_3$:**

$$w^\top x_3 + b = (-2)(1) + 4(1) + (-1)(2) + 0.5(0) - 1 = -2 + 4 - 2 + 0 - 1 = -1.$$

$$p(y = 1|x_3) = \sigma(-1) \approx 0.269 \Rightarrow \hat{y}_3 = 0$$

(incorrect, true label is 1).

**Prediction for $x_4$:**

$$w^\top x_4 + b = (-2)(1) + 4(0.5) + (-1)(-1) + 0.5(0) - 1 = -2 + 2 + 1 + 0 - 1 = 0.$$

$$p(y = 1|x_4) = \sigma(0) = 0.5 \Rightarrow \hat{y}_4 =?$$

(edge case).

**Performance.** Predicted labels:

if

$$\hat{y} = (1, 0, 0, 1), \qquad \text{true labels} = (1, 0, 1, 1).$$

Thus the model gets 3 out of 4 correct, i.e. 75% accuracy.

if

$$\hat{y} = (1, 0, 0, 0), \qquad \text{true labels} = (1, 0, 1, 1).$$

Thus the model gets 2 out of 4 correct, i.e. 50% accuracy.

## Learning

### Optimization

**Exercise 4.**
Given the following linear regression model and the following training dataset.

Model:

$$w = \begin{bmatrix} 3 \\ -1.5 \\ 2 \\ 0.5 \end{bmatrix}, \quad b = -1$$

Dataset:

$$(x_1 = \begin{bmatrix} 0 \\ -1 \\ 1 \\ 4 \end{bmatrix}, \quad y_1 = 3), \quad (x_2 = \begin{bmatrix} 2 \\ -1 \\ -2 \\ -1 \end{bmatrix}, \quad y_2 = -1)$$

(1) Calculate the mean squared error of the dataset.

(2) Compute the derivative of the error with respect to its weights $w$ and $b$.

(3) Perform one step of gradient descent with a learning rate of $\eta = 0.1$.

**Solution 4.**
**(1) Mean squared error.** For $\hat{y}_i = w^\top x_i + b$ and $e_i = \hat{y}_i - y_i$:

$$\hat{y}_1 = 3 \cdot 0 + (-1.5)(-1) + 2 \cdot 1 + 0.5 \cdot 4 - 1 = 4.5, \quad e_1 = 1.5,\ e_1^2 = 2.25,$$
$$\hat{y}_2 = 3 \cdot 2 + (-1.5)(-1) + 2 \cdot (-2) + 0.5 \cdot (-1) - 1 = 2.0, \quad e_2 = 3.0,\ e_2^2 = 9.$$

$$\text{MSE} = \frac{e_1^2 + e_2^2}{2} = \frac{2.25 + 9}{2} = \boxed{5.625}.$$

**(2) Gradients.** With MSE $E = \frac{1}{n}\sum_{i=1}^{2} e_i^2$, and $e_i = \hat{y}_i - y_i$ we have

$$\nabla_w E = \frac{2}{n}\sum_{i=1}^{2} e_i x_i \qquad \frac{\partial E}{\partial b} = \frac{2}{n}\sum_{i=1}^{2} e_i.$$

**Derivation of the gradient term $\nabla_w E = \frac{2}{n}\sum e_i x_i$**

We start from the definition of the mean squared error (MSE) loss:

$$E = \frac{1}{n}\sum_{i=1}^{n} e_i^2, \qquad \text{where } e_i = \hat{y}_i - y_i = (w^\top x_i + b) - y_i.$$

**Step 1. Substitute the definition of $e_i$:**

$$E = \frac{1}{n}\sum_{i=1}^{n}(w^\top x_i + b - y_i)^2.$$

We now compute the gradient with respect to the weight vector $w$:

$$\nabla_w E = \frac{\partial E}{\partial w}.$$

**Step 2. Differentiate inside the summation:** Since the derivative of a sum is the sum of derivatives,

$$\nabla_w E = \frac{1}{n}\sum_{i=1}^{n}\nabla_w\big((w^\top x_i + b - y_i)^2\big).$$

**Step 3. Apply the chain rule:** Let

$$z_i = w^\top x_i + b - y_i = e_i.$$

Then,

$$\nabla_w(z_i^2) = 2z_i\nabla_w z_i.$$

Because

$$\nabla_w z_i = \nabla_w(w^\top x_i + b - y_i) = x_i,$$

we obtain

$$\nabla_w(w^\top x_i + b - y_i)^2 = 2e_i x_i.$$

**Step 4. Substitute back into the summation:**

$$\nabla_w E = \frac{1}{n}\sum_{i=1}^{n}2e_i x_i = \boxed{\frac{2}{n}\sum_{i=1}^{n}e_i x_i.}$$

**Step 5. Substitute with $e_i$ and $x_i$:** Using $x_1 = \begin{bmatrix}0\\-1\\1\\4\end{bmatrix}$, $x_2 = \begin{bmatrix}2\\-1\\-2\\-1\end{bmatrix}$, $e_1 = 1.5$, $e_2 = 3$:

$$\sum e_i x_i = 1.5\begin{bmatrix}0\\-1\\1\\4\end{bmatrix} + 3\begin{bmatrix}2\\-1\\-2\\-1\end{bmatrix} = \begin{bmatrix}6\\-4.5\\-4.5\\3\end{bmatrix}, \quad \sum e_i = 1.5 + 3 = 4.5.$$

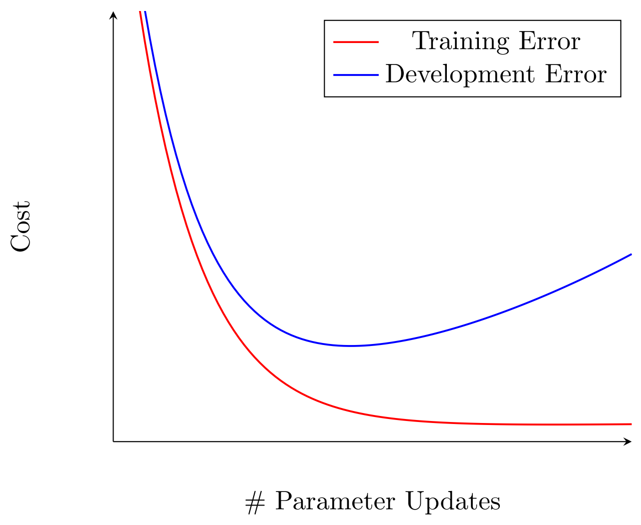
$$\boxed{\nabla_w E = \begin{bmatrix}6\\-4.5\\-4.5\\3\end{bmatrix}}, \qquad \boxed{\frac{\partial E}{\partial b} = 4.5}.$$

**(3) One GD step with $\eta = 0.1$.**

$$w^{\text{new}} = w - \eta\nabla_w E = \begin{bmatrix}3\\-1.5\\2\\0.5\end{bmatrix} - 0.1\begin{bmatrix}6\\-4.5\\-4.5\\3\end{bmatrix} = \boxed{\begin{bmatrix}2.4\\-1.05\\2.45\\0.2\end{bmatrix}}, \qquad b^{\text{new}} = b - \eta\frac{\partial E}{\partial b} = -1 - 0.1\cdot 4.5 = \boxed{-1.45}.$$

## Training curves

**Exercise 5.**
When training a ML model and monitoring its loss on the training and development data, the following curves are observed:



Do the curves behave as expected? Explain your answer.

**Solution 5.**
**Summary.** Yes, the curves behave as expected. Both training and development errors decrease at first, indicating effective learning. After some updates the development error starts rising while the training error continues to fall, which is the classic signature of overfitting (loss of generalization).

No, not as expected. We hope the development loss should always decrease.

**1) What the curves show.**

| **Curve** | **Color** | **Meaning** |
|---|---|---|
| Training error | red | Loss measured on the training set (the objective being optimized). |
| Development error | blue | Loss measured on held-out validation data (proxy for generalization). |

**2) Description of the behavior.**

- *Early phase:* both training and development errors decrease as the model starts capturing useful patterns.
- *Middle phase:* development error reaches a minimum while training error keeps decreasing; this is the best generalization point.
- *Later phase:* training error continues to decrease, but development error increases; the model overfits the training data.

## General questions

**Exercise 6.**
Mark whether the following statements are true or false and explain in short your solutions.

**Solution 6.**
**True/False statements and explanations.**

| **Statement** | **True** | **False** |
|---|---|---|
| Hyperparameters are tuned to optimize the results on the training set. | | |
| An overfitted model perfectly matches the development data. | | |
| Support vector machines can also handle multiclass classification tasks. | | |
| Decision tree classifiers are easy to interpret and visualize | | |
| In order to train LDA, we need labelled data. | | |

| **Statement** | **True** | **False** | **Explanation** |
|---|---|---|---|
| Hyperparameters are tuned to optimize the results on the training set. | | ✓ | Hyperparameters are tuned using the *validation (development)* set, not the training set, in order to estimate generalization performance. |
| An overfitted model perfectly matches the development data. | | ✓ | An overfitted model fits the *training* data extremely well but performs poorly on the development or test data. |
| Support vector machines can also handle multiclass classification tasks. | ✓ | | Although originally designed for binary classification, SVMs can be extended to multiclass problems using one-vs-one or one-vs-rest strategies. |
| Decision tree classifiers are easy to interpret and visualize. | ✓ | | Each decision path corresponds to explicit human-readable rules, so decision trees are considered highly interpretable. |
| In order to train LDA (Linear Discriminant Analysis), we need labelled data. | ✓ | | LDA is a supervised learning method; it requires class labels to compute between-class and within-class scatter matrices. |