Imperial College Business School **(2h)**

Machine Learning and Finance - Final Exam Solution -

Instructors: Arnaud de Servigny & Hachem Madmoun

Notations:

- For all $z \in \mathbb{R}^D$, the $\mathcal{L}^2$ norm on $\mathbb{R}^D$ of z is defined as follows: $||z||_2^2 = z^T z$
- The gradient of a function $f : \theta \in \mathbb{R}^D \mapsto \mathbb{R}$ at $\theta \in \mathbb{R}^D$ is denoted as follows $\nabla_\theta f(\theta) = \left(\frac{\partial f}{\partial \theta_1}(\theta), \ldots, \frac{\partial f}{\partial \theta_D}(\theta)\right)$
- $\mathcal{M}_{n,p}(\mathbb{R})$ is the space of the matrices composed of n rows and p columns.
- $I_n \in \mathcal{M}_{n,n}(\mathbb{R})$ is the identity matrix of size n.
- **Convention**: D dimensional row vectors are considered $\mathcal{M}_{D,1}(\mathbb{R})$ matrices.

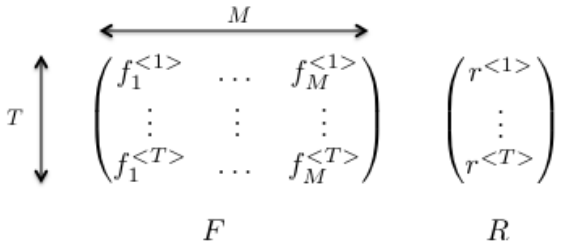
1 Building a factor model (60 marks)

Algorithmic trading strategies use factor models to quantify the relationship between the return of an asset and the sources of risk that are the main drivers of these returns.

We wish to predict the return of an asset based on M features called **factor premia**.

For each time step t in $\{1, \ldots, T\}$, the return of the asset is denoted $r^{<t>}$, and the M features are denoted $(f_i^{<t>})_{1 \leq i \leq M}$.

In the two following sections, the training data is composed of the **feature matrix** F of shape (T, M) and the output observation matrix R of shape $(T, 1)$:



1.1 Introducing a basic Regression Model

A **factor model** simply decompose the return of the asset at time t (denoted $r^{<t>}$) into the set of **factor premia** $(f_i^{<t>})_{1 \leq i \leq M}$ as follows:

$$\forall t \in \{1, \ldots, T\} \quad r^{<t>} = \sum_{i=1}^{M} \beta_i f_i^{<t>} + \alpha + \epsilon \text{ with } (\beta_1, \ldots, \beta_M, \alpha) \in \mathbb{R}^{M+1}, \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{1.1}$$

Question 1: What is the name of the model defined in equation 1.1 ? (3 marks)

Solution 1:

Equation 1.1 represents the **Linear Regression Model.**

Let $F^{<1>}, \dots, F^{<T>}$ be the rows of the matrix F and $\beta = (\beta_1, \dots, \beta_M, \alpha)^T \in \mathcal{M}_{M+1,1}(\mathbb{R})$ be the vector of all the parameters we want te estimate using the training data.

We want to re-write the equation 1.1 in a matrix form as follows:

$$\forall t \in \{1, \dots, T\} \quad r^{<t>} = \beta^T \tilde{F}^{<t>} + \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Question 2: Deduce $\tilde{\mathbf{F}}^{<\mathbf{t}>}$ from $\mathbf{F}^{<\mathbf{t}>}$ for all $\mathbf{t} \in \{1, \dots, T\}$. (3 marks)

Solution 2:

$\forall t \in \{1, \dots, T\} \quad \tilde{F}^{<t>} = [F^{<t>} \quad 1]^T \in \mathcal{M}_{M+1,1}(\mathbb{R})$

Let $\tilde{F}$ be the matrix composed of the rows $\tilde{F}^{<t>} \quad \forall t \in \{1, \dots, T\}$

Question 3: What is the shape of the $\tilde{\mathbf{F}}$ matrix ? (3 marks)

Solution 3:

$\tilde{F}$ is composed of T rows, each row is of dimension $M + 1$.
Thus, the shape of $\tilde{F}$ is $(T, M + 1)$

Question 4: Show that the optimal vector of parameters $\beta^* \in \mathcal{M}_{\mathbf{M+1,1}}(\mathbb{R})$ is defined as follows:

$$\beta^* = \underset{\beta \in \mathcal{M}_{M+1,1}(\mathbb{R})}{\arg\min} \frac{1}{T} ||\tilde{F}\beta - R||_2^2 \quad \text{(6 marks)} \tag{1.2}$$

Solution 4:

- The dataset is composed of i.i.d samples, the likelihood of the training dataset can be expressed as follows:

$$\begin{aligned} \mathcal{L}(\beta) &= \prod_{t=1}^{T} p(r^{<t>} | \tilde{F}^{<t>}; \beta) \\ &= \prod_{t=1}^{T} \left(\frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{1}{2} \frac{(r^{<t>} - \beta^T \tilde{F}^{<t>})^2}{\sigma^2}) \right) \end{aligned}$$

- Maximizing the likelihood is equivalent to minimizing the following negative log- likelihood:

$$\begin{aligned} -\log(\mathcal{L}(\beta)) &= -\sum_{t=1}^{T} \log(p(r^{<t>} | \tilde{F}^{<t>}; \beta)) \\ &= \frac{T}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{t=1}^{T} (r^{<t>} - \beta^T \tilde{F}^{<t>})^2 \end{aligned}$$

- The training problem can then be written as the following equivalent minimization problem:

$$\beta^* = \underset{\beta \in \mathcal{M}_{M+1,1}(\mathbb{R})}{\arg\min} \frac{1}{T} \sum_{t=1}^{T} (r^{<t>} - \beta^T \tilde{F}^{<t>})^2 = \underset{\beta \in \mathcal{M}_{M+1,1}(\mathbb{R})}{\arg\min} \frac{1}{T} ||\tilde{F}\beta - R||_2^2$$

The gradient of the function $J(\beta) := \frac{1}{T}\ ||\tilde{F}\beta - R||_2^2$ with respect to β is given in equation 1.3

$$\nabla_\beta J(\beta) = \frac{2}{T}(\tilde{F}^T\tilde{F}\beta - \tilde{F}^T R) \tag{1.3}$$

We would like to optimize the function J using the **Batch Gradient Descent Algorithm**.

Question 5: What is the expression of the loss on each batch ? Describe the optimization process. (5 marks)

Solution 5:

- We first decompose the training dataset into several batches (each batch is of size N_{batch}).
- Let $\tilde{F}_{\text{batch}}$ and R_{batch} be the matrices extracted from $\tilde{F}$ and R and containing the rows of a specific batch. Then, we define the loss over the batch as follows: $J_{\text{batch}}(\beta) = \frac{1}{N_{\text{batch}}}||\tilde{F}_{\text{batch}}\beta - R_{\text{batch}}||_2^2$
- The Batch Gradient Descent algorithm: For N_{epochs} epochs and η as a learning rate:
  - Initialize randomly β_0
  - Repeat N_{epochs} times:
    - For each batch in the set of batches, update the parameters:

$$\beta_{k+1} \leftarrow \beta_k - \eta\nabla_\beta J_{\text{batch}}(\beta_k)$$

The following figure shows the prediction error (the Root Mean Square Error denoted RMSE) for each epoch for the training and validation set.

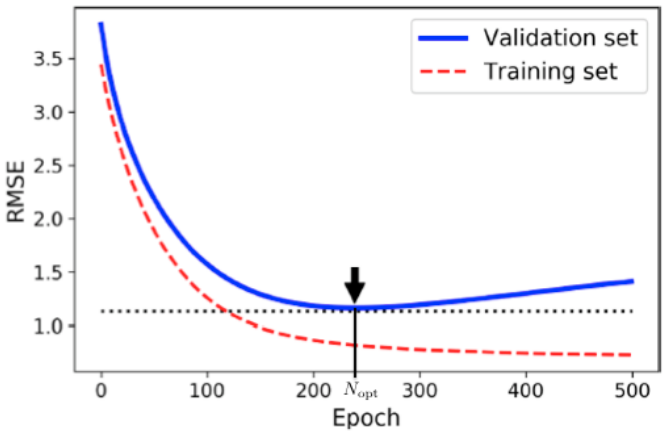


Figure 1: Training and validation error

Question 6: What is the problem highlighted in the the figure 1 and what should be the optimal number of epochs? (5 marks)

Solution 6:

- During the first phase (from the first epoch to N_{opt}), the algorithm is learning, its prediction error (RMSE) is decreasing for both the training and the validation set.
- After N_{opt}, the validation error stops decreasing and starts to go up. This indicates that the model has started to overfit the training data.
- An efficient and simple regularization technique is the **early stopping** method, which consists in stopping the training after N_{opt} epochs, before the model starts overfitting the training data.

1.2 Introducing regularization techniques to the Regression Model

An important theoratical result of Machine Learning is the fact that a model's generalization error can be expressed as the sum of the three following errors: the **bias**, the **variance** and the **irreducible error**.

Question 7: Explain the bias-variance tradeoff. (6 marks)

Solution 7:

- Increasing a model's complexity increases its variance and reduces its bias. It's called **overfitting**.
- Reducing a model's complexity increases its bias and reduces its variance. It's called **underfitting**
- Choosing the optimal complexity is called the bias-variance tradeoff, as shown in figure 2

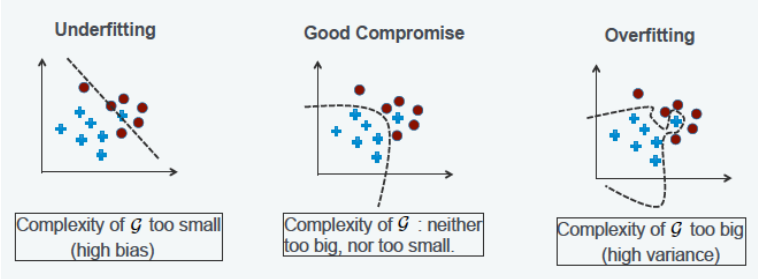


Figure 2: Choosing the optimal complexity

One popular approach to control the overfitting problem is that of **regularization**, which involves the addition of a penalty term to the error function to discourage the regression coeffictions from reaching large values.

The added penalty turns the optimal linear regression coefficients into the solution to the following minimization problem:

$$\beta^*_{\text{reg}} = \underset{\beta_{\text{reg}} \in \mathcal{M}_{M+1,1}(\mathbb{R})}{\arg\min} \frac{1}{T} \left(||\tilde{F}\beta_{\text{reg}} - R||_2^2 + \lambda \mathcal{S}(\beta_{\text{reg}}) \right) \tag{1.4}$$

These **shrinkage methods** differ by how they calculate the penalty term. The most common versions for the linear regression model are the **ridge regression** and the **lasso regression**.

In this section, let us consider the **ridge regression**. The penalty term is then defined as follows:

$$\mathcal{S}(\beta_{\text{reg}}) = \beta_{\text{reg}}^T \beta_{\text{reg}}$$

Equation 1.4 becomes:

$$\beta^*_{\text{reg}} = \underset{\beta_{\text{reg}} \in \mathcal{M}_{M+1,1}(\mathbb{R})}{\arg\min} \frac{1}{T} \left(||\tilde{F}\beta_{\text{reg}} - R||_2^2 + \lambda \beta_{\text{reg}}^T \beta_{\text{reg}} \right) \tag{1.5}$$

Question 8: Show that the closed solution to the ridge regression problem defined in 1.5 is:

$$\beta^*_{\text{reg}} = (\tilde{F}^T\tilde{F} + \lambda I_{M+1})^{-1}\tilde{F}^T R \quad \text{(8 marks)}$$
$$\text{(Hint : } \forall z \in \mathbb{R}^D \ \forall A \in \mathcal{M}_{D,D}(\mathbb{R}) \quad \nabla_z (z^T A z) = (A + A^T)z)$$

Solution 8:

- Let's consider the following loss function: $\tilde{J}(\beta_{\text{reg}}) = \frac{1}{T} \left(||\tilde{F}\beta_{\text{reg}} - R||_2^2 + \lambda \beta_{\text{reg}}^T \beta_{\text{reg}} \right)$
- Let's calculate the gradient of the loss function $\tilde{J}$:

$$\begin{aligned}
\nabla_{\beta_{\text{reg}}} \tilde{J}(\beta_{\text{reg}}) &= \nabla_{\beta_{\text{reg}}} \left(\frac{1}{T} \left(||\tilde{F}\beta_{\text{reg}} - R||_2^2 \right) \right) + \nabla_{\beta_{\text{reg}}} \left(\frac{\lambda}{T} \beta_{\text{reg}}^T \beta_{\text{reg}} \right) \\
&= \frac{2}{T}(\tilde{F}^T\tilde{F}\beta_{\text{reg}} - \tilde{F}^T R) + \frac{2\lambda}{T}\beta_{\text{reg}} \\
&= \frac{2}{T}(\tilde{F}^T\tilde{F} + \lambda I_{M+1})\beta_{\text{reg}} - \frac{2}{T}\tilde{F}^T R
\end{aligned}$$

- By setting the gradient to zero, we can solve the minimization problem 1.5:

$$\begin{aligned}
\nabla_{\beta_{\text{reg}}} \tilde{J}(\beta_{\text{reg}}) = 0 &\iff (\tilde{F}^T\tilde{F} + \lambda I_{M+1})\beta_{\text{reg}} = \tilde{F}^T R \\
&\iff \beta_{\text{reg}} = (\tilde{F}^T\tilde{F} + \lambda I_{M+1})^{-1}\tilde{F}^T R
\end{aligned}$$

Question 9: What would be the closed solution to the linear regression problem 1.2 ? (4 marks)

Solution 9:

- By setting $\lambda = 0$, we get the closed solution to the linear regression problem 1.2:

$$\beta^* = (\tilde{F}^T\tilde{F})^{-1}\tilde{F}^T R$$

Question 10: Why did we choose an iterative learning algorithm to solve the linear regression problem 1.2 and a closed solution for the ridge regression problem 1.5 ? (5 marks)

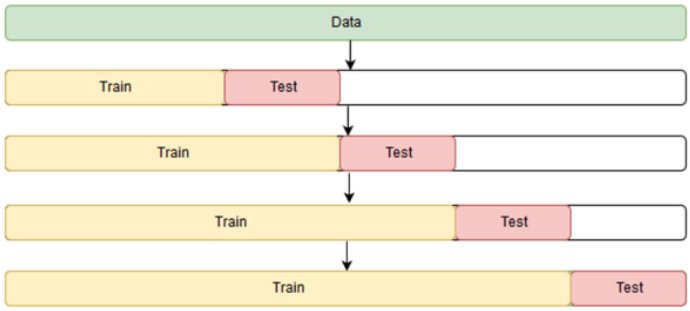
Solution 10:

- The solution to the ridge regression added λI_{M+1} to the matrix $\tilde{F}^T\tilde{F}$ before the inversion, which guarantees that the problem is non-singular, even if the matrix $\tilde{F}^T\tilde{F}$ does not have full rank.
- For the regular regression problem, we prefer an iterative learning algorithm to avoid having to invert the matrix $\tilde{F}^T\tilde{F}$.

Question 11: The hyperparameter λ controls the strength of the regularization. How would you choose the appropriate λ ? (6 marks)

Solution 11:

- The appropriate λ should be chosen using cross-validation for time series.
- We do not want to introduce look-ahead bias or leakage. Thus, after a test period is used, it becomes part of the training data that rolls forward as shown in the following figure:



Question 12: Describe three other methods used to prevent the overfitting problem in the context of Neural Networks or Tree Based Models. (6 marks)

Solution 12:

- **Dropout** for neural networks: It consists in randomly dropping out a number of output features of the layer during training.
- **Weight regularization** for neural networks: It consists in adding to the loss function a cost associated with having large weights. (The $\mathcal{L}^1$ norm of the weights or the $\mathcal{L}^2$ norm can be used as penaly terms).
- **Early stopping** for neural network: which consists in stopping the training process when the validation loss starts increasing.
- **Setting a maximum depth** for a decision tree algorithm. As decision tree algorithms are prone to overfitting. One way of handling the overfitting issue is to set a maximum depth for the tree.
- **Randomly select some attributes at each node split**: Another way of handling the overfitting problem for decision tree algorithms is used in the Random Forest algorithm, which is to maximize the information gain at each node split on a randomly selected subset of attributes.
- **Aggregating decision trees trained on different bootstrap samples**: Aggregating weak and decorrelated decision trees can reduce the variance of each of them and result in a strong learner.

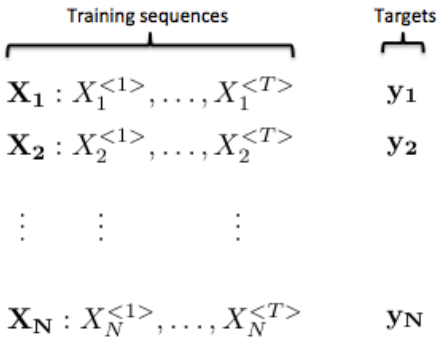
2 Building a Sentiment Analysis model (60 marks)

We wish to create a sentiment analysis model to classify financial news into three possible labels: **positive**, **negative** or **neutral**.

The training dataset is composed of N sentences $(X_i)_{1\leq i\leq N}$. Each sentence is composed of T words.

Let V be the vocabulary size. The first step of the processing consists in creating a dictionary to map each word to a discrete category in $\{1,\dots,V\}$.

We end up with N sequences $(X_i)_{1\leq i\leq N}$ of categories $X_i^{<t>}$ in $\{1,\dots,V\}$ for all $i \in \{1,\dots,N\}$ and for all $t \in \{1,\dots,T\}$ as shown in the following figure:



The three possible labels are encoded as follows: 0 for the negative sentiment, 1 for the neutral sentiment and 2 for the positive sentiment.

2.1 Using a Generative Classifier

In this second section, we would like to create a generative classifier. For that, we need to train three class conditional density functions, one for each target value $k \in \{0,1,2\}$

Each class conditional density function associated with the target $k \in \{0,1,2\}$ is parameterized by θ_k, which enables us to calculate $p_{\theta_k}(X|y=k)$ for a given a sequence $X = (X^{<1>},\dots,X^{<T>}) \in \{1,\dots,V\}^T$.

2.1.1 Predicting the target

Question 13: On which data are we going to train each class conditional discrete density function ? (3 marks)

Solution 13:

- Let's fix $k \in \{0,1,2\}$
- The class conditional discrete density estimator $p_{\theta_k}(.|y=k)$ is going to be trained on all the sequences associated with the target k.

Question 14: Let $\mathbf{X} = (\mathbf{X^{<1>}},\dots,\mathbf{X^{<T>}}) \in \{\mathbf{1},\dots,\mathbf{V}\}^{\mathbf{T}}$ be a new sequence. Express $\mathbf{p(y=k|X)}$ as a function of $(\mathbf{p(y=j)})_{\mathbf{j}\in\{0,1,2\}}$ and $(\mathbf{p}_{\theta_j}\mathbf{(X|y=j)})_{\mathbf{j}\in\{0,1,2\}}$(6 marks)

Solution 14:

- By using Bayes rule:

$$p(y=k|X) = \frac{p_{\theta_k}(X|y=k)p(y=k)}{\sum_{j=0}^{2} p_{\theta_j}(X|y=j)p(y=j)}$$

2.1.2 Using a Hidden Markov Model as a class conditional discrete density estimator on the discrete categories

Let k be in $\{0, 1, 2\}$. We would like to use a Hidden Markov Model (HMM) as a class conditional discrete density estimator $p_{\theta_k}(X|y = k)$ (where $X = (X^{<1>}, \ldots, X^{<T>}) \in \{1, \ldots, V\}^T$).

Let M be the number of hidden states.

Question 15: What are the parameters θ_k of the HMM class conditional discrete density estimator? (5 marks)

Solution 15:

- Each set of parameters θ_k is composed of the following matrices:
  - The initial vector $\pi_k \in \mathbb{R}^M$.
  - The transition matrix $Q_k \in \mathcal{M}_{M,M}(\mathbb{R})$.
  - The observation matrix $O_k \in \mathcal{M}_{M,V}(\mathbb{R})$

Question 16: By choosing reasonable values of V and M, compare the number of parameters of the HMM model with the number of parameters of a simple Markov Model (6 marks)

Solution 16:

- The number of parameters of the HMM model: $M + M^2 + M * V$
- The number of parameters of a simple Markov model: $V + V^2$
- Let's take $M = 20$ and $V = 100000$. We get 2000420 parameters for the HMM and 10000100000 parameters for the Markov model.
- The Markov model suffers from the curse of dimensionality.

Question 17: What training method can we use to estimate the parameters of the HMM? (3 marks)

Solution 17:

- There are two main algorithms for learning the parameters of the HMM model:
  - The Spectral Algorithm.
  - The Expectation Maximization Algorithm.

2.1.3 Using a Hidden Markov Model as a class conditional density estimator on the embedding vectors

Instead of modeling $p_{\theta_k}(X|y = k)$ for $X = (X^{<1>}, \ldots, X^{<T>}) \in \{1, \ldots, V\}^T$, we would like to consider instead a class conditional density estimator on the sequences of some embedding vectors.

Each category in $\{1, \ldots, V\}$ represents a word and can be mapped into a D-dimensional vector, where each dimension encodes part of the information about the corresponding word.

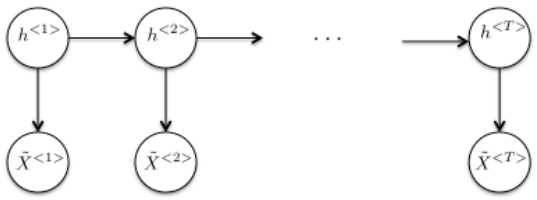
Question 18: Give two examples of unsupervised models used to create such embedding vectors and describe briefly one of them.(6 marks)

Solution 18:

- The **GloVe** algorithm, which consists of a weighted least squares model that trains on global word-word co-occurence counts.
- The **Wod2vec** algorithms, a window based model, which learns word embeddings by making predictions in local context windows. The intuition behind the model is summarized by the famous quote "You shall know a word by the company it keeps". The model demonstrates the capacity to capture complex linguistic patters beyond word similarity.

For each new sequence $X = (X^{<1>}, \dots, X^{<T>}) \in \{1, \dots, V\}^T$, we define $\tilde{X} = (\tilde{X}^{<1>}, \dots, \tilde{X}^{<T>})$ the sequence of the embedding vectors $\tilde{X}^{<t>} \in \mathbb{R}^D$ associated with the categories $X^{<t>} \in \{1, \dots, V\}$ for each $t \in \{1, \dots, T\}$.

We would like to use an HMM model in order to learn the distribution of a sequence of D-dimensional embedding vectors $\tilde{X} = (\tilde{X}^{<1>}, \dots, \tilde{X}^{<T>})$ as represented in the following figure:



For each $t \in \{1, \dots, T\}$, the embeding vector $\tilde{X}^{<t>} \in \mathbb{R}^D$ is associated with a hidden state $h^{<t>} \in \{1, \dots, M\}$.

Let m be in $\{1, \dots, M\}$. The emission probability distribution of the observation $X^{<t>}$ conditioned on the hidden state $h^{<t>} = m$ is parameterized by a multivariate normal distribution with a mean vector $\mu_m \in \mathbb{R}^D$ and a covariance matrix $\Sigma_m \in \mathcal{M}_{D,D}(\mathbb{R})$, as explained in equation 2.1

$$\forall t \in \{1, \dots, T\} \; \forall m \in \{1, \dots, M\} \quad X^{<t>} | h^{<t>} = m \sim \mathcal{N}_D(\mu_m, \Sigma_m) \tag{2.1}$$

Let $\tilde{\theta_k}$ be the set of the parameters of the HMM class conditional density estimator $p_{\tilde{\theta}_k}(\tilde{X}|y = k)$ (where $\tilde{X}$ is a sequence of embedding vectors $(\tilde{X}^{<1>}, \dots, \tilde{X}^{<T>})$

Question 19: What are the parameters $\tilde{\theta_k}$ of the HMM class conditional density estimator on the embedding vectors? (6 marks)

Solution 19:

- Each set of parameters $\tilde{\theta_k}$ is composed of the following matrices:
  - The initial vector $\pi_k \in \mathbb{R}^M$
  - The transition matrix $Q_k \in \mathcal{M}_{M,M}(\mathbb{R})$
  - For each hidden state $m \in \{1, \dots, M\}$, the emission probability is a multivariate normal distribution parameterized by the following matrices:
    - The mean vector $\mu_m \in \mathbb{R}^D$
    - The covariance matrix $\Sigma_m \in \mathcal{M}_{D,D}(\mathbb{R})$

Question 20: By choosing reasonable values for M, D and V, compare the number of parameters of the HMM model on the discrete observations in $\{1, \dots, V\}$ with the number of parameters of the HMM on the D-dimensional embedding vectors. (6 marks)

Solution 20:

- For the HMM model on the discrete observations, we have $M + M^2 + M * V$ parameters.
- For the HMM model on the embedding vectors, we have $M + M^2 + M * (D + D^2)$ parameters.
- With $V = 100000$, $M = 20$, and $D = 100$, the HMM on the discrete observations has 2000420 parameters and the HMM on the continuous observations has 202420
- We have much less parameters to learn with the continuous embedding vectors.

2.2 Using a Sequential Neural Network

We would like to use a **Sequential Neural Network** model with LSTM layers and pre-trained D-dimensional word vectors to perform the same classification of news.

Question 21: Define such a model by specifying the different layers, the different hyperparameters for each layer and how the shape of the data is changing after each layer transformation. (10 marks)

Solution 21:

- A proposed model: The sequence of the following layers:
  - The tensor of data is of shape (N, T). It contains the sequences of integers representing the words.
  - The first layer is an **Embedding layer**. It will transform the (N, T) tensor into an (N, T, D) tensor. Each integer (representing a word) will be encoded into a D-dimensional vector. As we want to use pretrainded word vectors, we dont want the embedding matrix to be updated during the training process. So, we will load a pretrained embedding matrix (like the one trained on the Wikipedia data using the Word2vec algorithm).
  - We can then stack p **LSTM layers**. Each of the p LSTM layers will return the sequence of outputs. Hence, the (N, T, D) tensor will be transformed into an (N, T, d_1) tensor, then (N, T, d_2), until the output of the last LSTM layer: an (N, T, d_p) tensor.
  - We can then use a last LSTM layer, which only returns the last output: a vector of size d. We get a tensor of shape (N, d)
  - As we are dealing with a multiclass classification problem with $K = 3$ categories, the last layer is a **dense layer** with 3 neurons and a softmax activation function. The final tensor is then of shape (N, K)

Question 22: What loss function should we use? (3 marks)

Solution 22:

- For the multiclass classification problem, we can use the categorical crossentropy loss function.

Question 23: Describe an appropriate optimizer with an adaptive learning rate. (6 marks)

Solution 23:

- We can use the Adam optimizer (Adaptive Moment Estimation), which computes adaptive learning rates for each parameter.
- In addition to storing an exponentially decaying average of past squared gradients v_t, Adam also keeps an exponentially decaying average of past gradients m_t

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

- As m_t and v_t are initialized as vectors of zeros, they are biased towards zero during the initial time steps.
- Thus, we use the bias-corrected first and second moment estimates:

$$\tilde{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\tilde{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- We fix the learning rate η
- The Adam update equation of the parameters θ is then:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\tilde{v}_t} + \epsilon} \tilde{m}_t$$

- $\epsilon = 10^{-8}$ is a very small value used to avoid dividing by zero.
- We usually use $\beta_1 = 0.9$ and $\beta_2 = 0.999$.
- We can see that in the update equation, the learning rate depends on $\tilde{v}_t$.