Advanced Methods in Text Analytics
Exercise 1: Machine Learning Basics

Daniel Ruffinelli

FSS 2025

The following is some of the mathematical notation we will use throughout the semester:

- Scalars: $a, b \in \mathbb{R}$
- Vectors: $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ (by default, vectors are all *column* vectors, i.e. size is $n \times 1$)
- Matrices: $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{m \times n}$
- Scalar product: $a\boldsymbol{x} \in \mathbb{R}^n$
- Dot product: $\boldsymbol{x}^\top \boldsymbol{y} \in \mathbb{R}$ (some sources call this a scalar product)
- Matrix-vector product: $\boldsymbol{X}\boldsymbol{x} \in \mathbb{R}^{m \times 1}$
- Matrix-matrix product: $\boldsymbol{X}^\top \boldsymbol{Y} \in \mathbb{R}^{n \times n}$

1 Machine Learning Basics

In machine learning, we typically develop models that can be trained to perform a specific task. Formally, a model can be represented by a function $f_{\boldsymbol{\theta}} \colon \mathbb{R}^n \mapsto \mathbb{R}^m$, where $\boldsymbol{\theta} \in \mathbb{R}^k$ are the parameters of the model. For a given $\boldsymbol{x} \in \mathbb{R}^n$, we can write this as follows: $f(\boldsymbol{x} \mid \boldsymbol{\theta}) = \boldsymbol{y}$.

(a) What part of the basic machine learning pipeline do the symbols in the formal description above denote? Specifically:

  (i) What does $\boldsymbol{x}$ represent? And each of $\boldsymbol{x}_i$?

  (ii) What is $\boldsymbol{y}$ and what is it used for?

  (iii) What is $f(\boldsymbol{x} \mid \boldsymbol{\theta})$? Explain this notation.

  (iv) What is a possible value for the input dimension n? And for the output dimension m? What factors determine these values?

(b) What are the common types of machine learning tasks in natural language processing based on the output a model produces? What does the output look like for each of these task types?

(c) How does *"learning"* take place in machine learning? What is the goal of this learning process?

(d) What are the *hyperparameters* of a machine learning model? Provide an example. How do we find the optimal values for a model's hyperparameters?

(e) Why do machine learning pipelines typically rely on datasets split into three parts: training, validation, and test sets? What is the role of each of these data splits?

(f) Briefly explain the concepts of *underfitting* and *overfitting*. Why do they often occur? How may we recognize them in practice?

2 Log-linear Classifiers

Linear classifiers are those that model the classification task as a *linear combination* of input features and model parameters, where the model parameters act as the *weights* of that linear combination. In this task, we review some log-linear classifiers commonly used in NLP.

(a) Recall that the logistic regression model is a linear classifier designed for binary classification. We introduced this model in the lectures with the following equations:

$$p(y = 1 \mid x) = \sigma(wx + b) \tag{1}$$

$$p(y = 0 \mid x) = 1 - \sigma(wx + b) \tag{2}$$

where $x, y, w, b \in \mathbb{R}$, $\boldsymbol{\theta} = (w, b)$ are model parameters, x is the input and σ is the logistic (sigmoid) function defined as follows:

$$\sigma(z) = \frac{\exp(z)}{1 + \exp(z)} \tag{3}$$

Give a formal expression for a logistic regression model that takes as input a vector $\boldsymbol{x} \in \mathbb{R}^n$ and that does not use a bias term. How many parameters does this model have?

(b) If not done already, write the same logistic regression model from the previous question but in vectorized form, i.e. as a function of its parameter vector $\boldsymbol{w}$.

(c) Now add a bias term to your logistic regression model. Can we still write it in vectorized form?

(d) *Log*-linear models are those that model classification as a linear combination of $\boldsymbol{x}$ and $\boldsymbol{w}$ with the exponential function applied to it. Give a formal expression for a *general* log-linear model $f(y \mid \boldsymbol{x})$ designed for binary classification and parameterized by $\boldsymbol{w}$, where $\boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^n$.

(e) Now generalize your answer from the previous question so the model is designed for multi-class classification. How many parameters does this model have?

(f) If not done already, rewrite your model from the previous question so it's a probabilistic model, i.e. for each input vector $\boldsymbol{x}$ and class c, the models provides the probability that the input belongs to that given class c. Can you do this in vectorized form? I.e. can you define a function that returns a distribution over classes for a given input vector $\boldsymbol{x}$?

(g) Let's put it all together now. Given $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ that represents m examples of n dimensions, write a formal expression for a general log-linear model designed for multi-class classification over C classes *in vectorized form.* Assume no biases.

(h) Can you guess why we use *log*-probabilities in log-linear models instead of just probabilities as in linear classifiers?

3 Training Log-linear Classifiers

In this task, we are interested in training a probabilistic log-linear model, like those from the previous task, using maximum likelihood estimation (MLE).

(a) In MLE, we aim to maximize the likelihood of a model given some data. Let $X = \{x_1, x_2, \dots, x_n\}$ be a dataset of n examples and p a probabilistic log-linear model parameterized by $\boldsymbol{w} \in \mathbb{R}^d$ and designed for binary classification. Given the following joint distribution of the data as given by the model:

$$p(X \mid \boldsymbol{w}) = p(x_1, x_2, \dots, x_n), \tag{4}$$

rewrite this expression assuming that all examples are independent and identically distributed (i.i.d.).

(b) For clarity, let X_{pos} be the set examples from the positive class, and X_{neg} the set of examples from the negative class. Rewrite your answer from the previous question as a function of X_{pos} and X_{neg}.

(c) Write the corresponding likelihood function $\mathcal{L}$ for the model p described in your previous answer.

(d) For reasons discussed in the previous task, we often work with log-probabilities instead of probabilities. Write the log-likelihood function $\mathcal{LL}$ for the model p. Use ln throughout.

(e) Assume p is a logistic regression model and each data point in X is represented by $\boldsymbol{x}_i \in \mathbb{R}^d$. Rewrite the log-likelihood function you wrote in the previous answer by including the logistic regression model explicitly, i.e. using the sigmoid function in Eq. 3.

(f) As seen in the MLE example from the lecture, we will need to compute the gradient to maximize our expression. Compute the gradient of the likelihood function for your logistic regression model w.r.t. model parameters $\boldsymbol{w}$. That is, $\nabla_{\boldsymbol{w}}\mathcal{LL}$ where ∇ is defined as:

$$\nabla_{\boldsymbol{w}}\mathcal{LL} = \left[\frac{\partial \mathcal{LL}}{\partial w_1}, \frac{\partial \mathcal{LL}}{\partial w_2}, \dots, \frac{\partial \mathcal{LL}}{\partial w_d}\right]. \tag{5}$$

(g) In most cases, there are no closed-form solutions to the MLE problem. Instead, we use gradient-based optimization methods. Recall the gradient descent (GD) method described in the following algorithm:

Algorithm 1 Gradient Descent

```
procedure GRADIENT DESCENT(Dataset $X$, Objective function $J$, Model $p_{\boldsymbol{w}}$, Learning rate $lr$, Number of Epochs $N$)
    $\boldsymbol{w} \leftarrow random()$        // random weight initialization
    for $i \in \{1, \dots N\}$ do
        $g_i \leftarrow \nabla_{\boldsymbol{w}} J(p_{\boldsymbol{w}}(X))$
        $\boldsymbol{w}_{i+1} \leftarrow \boldsymbol{w}_i - lr \cdot g_i$        // update rule
    end for
    Return $p_{\boldsymbol{w}}$
end procedure
```

When relating this algorithm to the previous questions, $p_{\boldsymbol{w}}$ is a logistic regression model, J is given by the answer to question (e), and g is given by Eq. 5. Modify Algorithm 1 so it performs (i) stochastic gradient descent (SGD) and (ii) mini-batch gradient descent (BGD). Recall that the former uses a single example for each update, and the latter a subset of the data for each update. Make sure you define each new function or component you use in the algorithm.

(h) Assume that the function shuffle(X) randomly shuffles the examples in dataset X. How would you modify your BGD algorithm to include this step? Do you know why shuffling is important?