Advanced Methods in Text Analytics

Exercise 01: ML Basics

Daniel Ruffinelli

University of Mannheim

FSS 2025

About These Tutorials

- **Goals**
  - *Reinforce/deepen* content from lectures
  - *Introduce new concepts* to support lecture content
  - Get some hands-on experience with *code*
- **Tutorial slides**
  - These are the type of slides I'll use for tutorials
  - They are mostly a copy of exercise sheets in slide format
  - I often make notes on them during tutorials
  - I can't guarantee notes are clear without being here
  - I also want to promote that students come to tutorials
  - So, **I will not make these slides publicly available**

Mathematical Notation

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: $\boldsymbol{X}^\top \boldsymbol{Y} \in \mathbb{R}^{n \times n}$

Machine Learning Basics

Context

- 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: $f(\boldsymbol{x} \mid \boldsymbol{\theta}) = \boldsymbol{y}$.

Machine Learning Basics

Question 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?

Machine Learning Basics

Answer a)

(i) $\boldsymbol{x}$ is input vector, typically represents point in some dataset. Input vectors often referred to as *feature vectors*, their corresponding vector space $\mathbb{R}^n$ as a *feature space*. These names derived from each $\boldsymbol{x}_i$ being referred to as a *feature*.

(ii) Output of our function, used for inference or to make predictions with the model.

(iii) This is read as "function $\boldsymbol{h}$ of $\boldsymbol{x}$ *given* parameters $\boldsymbol{\theta}$".

(iv) n can take any value, typically in $\mathbb{N}$, this is typically a design decision, e.g. number of features we manually crafted, number of PCA components used for dimensionality reduction, the size of each hidden layer in a feed-forward neural networm (FNN). m can also take any value in $\mathbb{N}$, i.e. $\boldsymbol{y} \in \mathbb{R}^d$, i.e. d not necessarily equal to n, less a design decision, more a property of the task we are modeling. E.g. for binary classification, we typically have $m = 1$, for classification problem with k classes, $m = k$.

Machine Learning Basics

Question b)

- **Question:**
  - 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?

Machine Learning Basics

Question b)

- **Question:**
  - 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?
- **Answer:**
  - **Classification:** output is categorical, e.g. part-of-speech (POS) tagging or next-word prediction.
  - **Regression:** output is a real number, e.g. in automated essay scoring.

Machine Learning Basics

Question c)

- **Question:**
  - How does **“learning”** take place in machine learning?
  - What is the **goal** of this learning process?

Machine Learning Basics
Question c)

- **Question:**
  - How does **"learning"** take place in machine learning?
  - What is the **goal** of this learning process?
- **Answer:**
  - **Learning:** estimating parameters $\boldsymbol{\theta}$ of function f in order to solve an optimization problem, usually minimizing some training objective
  - **Goal:** not to solve this optimization problem, but to perform well on unseen data, i.e. to perform a task well *generally*.

Machine Learning Basics

Question d)

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

Machine Learning Basics
Question d)

- **Question:**
  - What are the *hyperparameters* of a machine learning model? Provide an example.
  - How do we find the optimal values for a model's hyperparameters?
- **Answer:**
  - Parameters that we do not learn, but manually set.
  - These are often not parameters of the model, but they related to the training setting.
  - For example:
    - Using a bias or not
    - The choice of a loss function
    - The choice of a model
  - We typically find useful values for hyperparameters by conducting grid search or random search, although more involved methods exist.

Machine Learning Basics

Question 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?

Machine Learning Basics

Answer e)

- Related to goal of ML: models should generalize well.
- Datasets usually split into **training** and **test** splits, so we may estimate parameters of our models on training data, them evaluate our models on unseen data using the test set.
- But we make decisions before training (hyperparameters)
- Generally, we want know how these decisions impact our model, but when comparing different hyperparameter settings on test data, we would identify which one works best on *that* test set, which would make it no longer unseen.
- **Validation** split exists to allow for model selection without compromising test data, i.e. it's evaluation data we can use to train various models under different settings and choose the settings that give the best performace.
- To ensure all splits come from same distribution, splits constructed by shuffling a dataset. Then for each split, randomly sampling examples without replacement.

Machine Learning Basics

Question f)

- **Question:**
  - Briefly explain the concepts of **underfitting** and **overfitting**.
  - Why do they often occur?
  - How may we recognize them in practice?

Machine Learning Basics

Question f)

- **Question:**
  - Briefly explain the concepts of **underfitting** and **overfitting**.
  - Why do they often occur?
  - How may we recognize them in practice?
- **Answer:**
  - **Underfitting**: model is too simple to perform well on a task, even on the data it is trained on.
  - **Overfitting:** model is capable of performing task well to the point it learns all the detail/noise in training data, while still not performing well on unseen data
  - We can *measure underfitting* by looking at the model's performance on the training set, e.g. compute accuracy on training set in a classification setting
  - We can **measure overfitting** by looking at model performance on unseen data, e.g. a validation split, and contrasting it with its performance on training data.
  - High performance during training but low performance on unseen data: *clear indication of overfitting*, often not so straightforward, this gap is common, baselines required

Log-linear Classifiers

Context

- 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.

Log-linear Classifiers

Question 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?

Log-linear Classifiers

Answer a)

- Given that logistic regression is a linear classifier, a logistic regression model that takes an n-dimensional vector as input, and does not use a bias, can be formally described as follows:

$$p(y = 1 \mid x) = \sigma\left(\sum_{i=1}^{n} w_i \cdot x_i\right) \tag{4}$$

$$p(y = 0 \mid x) = 1 - p(y = 1), \tag{5}$$

  where $w_i \in \mathbb{R}$ is the parameter (weight) that corresponds to input feature x_i.

- Being a linear combination of the input features, this model has as many parameters as there are features in the input vector.

Log-linear Classifiers

Question b)

- **Question:**
  - 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 $\boldsymbol{w}$.

Log-linear Classifiers

Question b)

- **Question:**
  - 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 $\boldsymbol{w}$.
- **Answer:**
  - We have:

$$p(y = 1 \mid x) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}). \tag{6}$$

Log-linear Classifiers

Question c)

- **Question:**
  - Now add a bias term to your logistic regression model.
  - Can we still write it in vectorized form?

Log-linear Classifiers

Question c)

- **Question:**
  - Now add a bias term to your logistic regression model.
  - Can we still write it in vectorized form?
- **Answer:**
  - We have:

$$p(y = 1 \mid x) = \sigma\left(\sum_{i=1}^{n}(w_i \cdot x_i) + b\right) \tag{7}$$
$$p(y = 0 \mid x) = 1 - p(y = 1), \tag{8}$$

    where $b \in \mathbb{R}$ is the bias term.
  - We can still write this in vectorized form, but to do this comfortably we make use of what we call a *bias feature* $x_0 = 1$, which acts as the coefficient for the bias weight in the linear combination. That is:

$$p(y = 1 \mid x) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}). \tag{9}$$

    where $\boldsymbol{x} \in \mathbb{R}^{n+1}$, i.e. $\boldsymbol{x} = (x_o, x_1, \ldots, x_n)$.

Log-linear Classifiers

Question d)

- **Question:**
  - *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 $\boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^n$.

Log-linear Classifiers

Question d)

- **Question:**
  - *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 $\boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^n$.
- **Answer:**
  - We have:

$$f(y = 1 \mid \boldsymbol{x}) = \exp(\boldsymbol{w}^\top \boldsymbol{x}) = e^{\boldsymbol{w}^\top \boldsymbol{x}}, \tag{10}$$

$$f(y = 0 \mid \boldsymbol{x}) = 1 - f(y = 1 \mid \boldsymbol{x}). \tag{11}$$

Log-linear Classifiers

Question e)

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

Log-linear Classifiers

Question e)

- **Question:**
  - Now generalize your answer from the previous question so the model is designed for *multi-class* classification.
  - How many parameters does this model have?
- **Answer:**
  - For $c \in C$ classes, we have:

$$f(y = c \mid \boldsymbol{x}) = \exp(\boldsymbol{w}_c^\top \boldsymbol{x}) = e^{\boldsymbol{w}_c^\top \boldsymbol{x}}, \qquad (12)$$

    where $\boldsymbol{w}_c$ are the parameters used for making predictions for class c. That is, this general log-linear model has as many parameters as there are input features *for each class* $c \in C$.

Log-linear Classifiers

Question 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 $\boldsymbol{x}$?

Log-linear Classifiers

Answer f) (1)

- We can accomplish this by normalizing the output of the model, so that it represents a probability distribution over the classes.
- Concretely, we have:

$$p(y = c \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_c^\top \boldsymbol{x})}{\sum_{c' \in C} \exp(\boldsymbol{w}_{c'}^\top \boldsymbol{x})}, \tag{13}$$

  where C is the set of classes in our classification task.
- Eq. 13 is known as a *softmax function*, which gives the name to this multi-class log-linear model: softmax regression.
- Note that given input vector $\boldsymbol{x}$ and class c, the function returns a single probability value.
- That means, for a given input vector $\boldsymbol{x}$, we can get an entire “softmax distribution” over classes by computing this function for all classes $c \in C$.

Log-linear Classifiers
Answer f) (2)

- Then, by normalizing each entry in that vector as with Eq. 13, we get a probability vector $\boldsymbol{y} \in \mathbb{R}^C$ that contains the distribution over all classes, i.e. all of its entries add up to 1.
- That is:

$$\boldsymbol{y} = softmax(\boldsymbol{l}), \tag{14}$$

where the *i*-th component of $\boldsymbol{y}$ is given by:

$$\boldsymbol{y}_i = \frac{e^{l_i}}{\sum_{j=1}^{C} e^{l_j}}. \tag{15}$$

- This is typically the way the softmax function is represented in vectorized form, a very interpretable output

Log-linear Classifiers

Question g)

- **Question:**
  - 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).

Log-linear Classifiers

Question g)

- **Question:**
  - 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).
- **Answer:**
  - Let $\boldsymbol{W} \in \mathbb{R}^{n \times C}$ be parameters of our model such that column $\boldsymbol{w}_i$ corresponds to class $i \in C$.
  - Using our vectorized softmax function from the previous question, we have:

$$\boldsymbol{Y} = softmax(\boldsymbol{X}\boldsymbol{W}), \tag{16}$$

where we assume our softmax operation is applied row-wise.

Log-linear Classifiers

Question h)

- **Question:**
  - Can you guess why we use *log*-probabilities in log-linear models instead of just probabilities as in linear classifiers?

Log-linear Classifiers

Question h)

- **Question:**
  - Can you guess why we use *log*-probabilities in log-linear models instead of just probabilities as in linear classifiers?
- **Answer:**
  - There are a few reasons for this, but in short, there are some computational reasons, and some statistical reasons.
  - The computational reasons are (i) it simplifies the math and therefore computations, e.g. when computing likehoods of multiple data points, and (ii) it helps to avoid numerical underflow issues that can occur when multiplying many probabilities together.
  - The main statistical reason is that it makes the model more interpretable, as it can be related to concepts like odds and independence.

Training Log-linear Classifiers

Context

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

Training Log-linear Classifiers

Question a)

- **Question:**
  - In MLE, we aim to maximize the likelihood of a model given some data.
  - Let $X = \{x_1, x_2, \ldots, x_n\}$ 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:

$$p(X \mid \boldsymbol{w}) = p(x_1, x_2, \ldots, x_n), \tag{17}$$

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

Training Log-linear Classifiers

Question a)

- **Question:**
  - In MLE, we aim to maximize the likelihood of a model given some data.
  - Let $X = \{x_1, x_2, \ldots, x_n\}$ 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:

$$p(X \mid \boldsymbol{w}) = p(x_1, x_2, \ldots, x_n), \tag{17}$$

    rewrite this expression assuming that all examples are independent and identically distributed (i.i.d.).
- **Answer:**
  - We have:

$$p(X \mid \boldsymbol{w}) = p(x_1 \mid \boldsymbol{w}) p(x_2 \mid \boldsymbol{w}) \ldots p(x_n \mid \boldsymbol{w}) = \prod_{i=1}^{n} p(x_i \mid \boldsymbol{w}). \tag{18}$$

Training Log-linear Classifiers

Question b)

- **Question:**
  - 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}.

Training Log-linear Classifiers

Question b)

- **Question:**
  - 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}.
- **Answer:**
  - We have:

$$p(X \mid \boldsymbol{w}) = \prod_{x_p \in X_{pos}} p(x_p \mid \boldsymbol{w}) \prod_{x_n \in X_{neg}} (1 - p(x_n \mid \boldsymbol{w})) \,. \qquad (19)$$

Training Log-linear Classifiers

Question c)

- **Question:**
  - Write the corresponding likelihood function $\mathcal{L}$ for the model p described in your previous answer.

Training Log-linear Classifiers

Question c)

- **Question:**
  - Write the corresponding likelihood function $\mathcal{L}$ for the model p described in your previous answer.
- **Answer:**
  - We have:

$$\mathcal{L}(\boldsymbol{w} \mid X) = \prod_{x_p \in X_{pos}} p(x_p \mid \boldsymbol{w}) \prod_{x_n \in X_{neg}} (1 - p(x_n \mid \boldsymbol{w}))\,. \qquad (20)$$

Training Log-linear Classifiers

Question 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.

Training Log-linear Classifiers

Answer d)

- We have:

$$\mathcal{LL}(\boldsymbol{w} \mid X) = \ln p(X \mid \boldsymbol{w}) \tag{21}$$

$$= \ln \left(\prod_{i=1}^{n} p(x_i \mid \boldsymbol{w}) \right) \tag{22}$$

$$= \ln \left(\prod_{x_p \in X_{pos}} p(x_p \mid \boldsymbol{w}) \prod_{x_n \in X_{neg}} (1 - p(x_n \mid \boldsymbol{w})) \right) \tag{23}$$

$$= \sum_{x_p \in X_{pos}} \ln p(x_p \mid \boldsymbol{w}) + \sum_{x_n \in X_{neg}} \ln (1 - p(x_n \mid \boldsymbol{w})) . \tag{24}$$

- This is the general expression we maximize when training log-linear models with MLE using the i.i.d. assumption.
- We want to find value of $\boldsymbol{w}$ that maximizes Eq. 24.

Training Log-linear Classifiers

Question 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.

Training Log-linear Classifiers

Answer e)

- We have:

$$\mathcal{LL}(\boldsymbol{w} \mid X) = \sum_{x_p \in X_{pos}} \ln \sigma(\boldsymbol{w}^\top \boldsymbol{x}) + \sum_{x_n \in X_{neg}} \ln \left(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x})\right) \tag{25}$$

$$= \sum_{x_p \in X_{pos}} \ln \frac{e^{\boldsymbol{w}^\top \boldsymbol{x}_p}}{1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_p}} + \sum_{x_n \in X_{neg}} \ln \left(1 - \frac{e^{\boldsymbol{w}^\top \boldsymbol{x}_n}}{1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_n}}\right) \tag{26}$$

$$= \sum_{x_p \in X_{pos}} \ln \frac{e^{\boldsymbol{w}^\top \boldsymbol{x}_p}}{1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_p}} + \sum_{x_n \in X_{neg}} \ln \left(\frac{1}{1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_n}}\right) \tag{27}$$

$$= \sum_{x_p \in X_{pos}} \left(\boldsymbol{w}^\top \boldsymbol{x}_p - \ln(1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_p})\right) - \sum_{x_n \in X_{neg}} \ln \left(1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_n}\right). \tag{28}$$

Training Log-linear Classifiers
Question 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:

$$\nabla_{\boldsymbol{w}}\mathcal{LL} = \left[\frac{\partial\mathcal{LL}}{\partial w_1}, \frac{\partial\mathcal{LL}}{\partial w_2}, \ldots, \frac{\partial\mathcal{LL}}{\partial w_d}\right]. \tag{29}$$

Training Log-linear Classifiers

Answer f)

- Let $\nabla = \nabla_{\boldsymbol{w}}$. We have:

$$\nabla \mathcal{LL} = \nabla \left(\sum_{x_p \in X_{pos}} \left(\boldsymbol{w}^\top \boldsymbol{x}_p - \ln(1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_p}) \right) - \sum_{x_n \in X_{neg}} \ln \left(1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_n} \right) \right) \tag{30}$$

$$= \nabla \sum_{x_p \in X_{pos}} \left(\boldsymbol{w}^\top \boldsymbol{x}_p - \ln(1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_p}) \right) - \nabla \sum_{x_n \in X_{neg}} \ln \left(1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_n} \right) \tag{31}$$

$$= \sum_{x_p \in X_{pos}} \left(\boldsymbol{x}_p - \nabla \ln(1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_p}) \right) - \sum_{x_n \in X_{neg}} \nabla \ln \left(1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_n} \right) \tag{32}$$

$$= \sum_{x_p \in X_{pos}} \left(\boldsymbol{x}_p - \frac{e^{\boldsymbol{w}^\top \boldsymbol{x}_p}}{1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_p}} \boldsymbol{x}_p \right) - \sum_{x_n \in X_{neg}} \left(\frac{e^{\boldsymbol{w}^\top \boldsymbol{x}_n}}{1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_n}} \boldsymbol{x}_n \right) \tag{33}$$

$$= \sum_{x_p \in X_{pos}} \left(1 - \frac{e^{\boldsymbol{w}^\top \boldsymbol{x}_p}}{1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_p}} \right) \boldsymbol{x}_p - \sum_{x_n \in X_{neg}} \left(\frac{e^{\boldsymbol{w}^\top \boldsymbol{x}_n}}{1 + e^{\boldsymbol{w}^\top \boldsymbol{x}_n}} \boldsymbol{x}_n \right) \tag{34}$$

$$= \sum_{x_p \in X_{pos}} \left(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_p) \right) \boldsymbol{x}_p - \sum_{x_n \in X_{neg}} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) \boldsymbol{x}_n . \tag{35}$$

Training Log-linear Classifiers

Question g) (1)

- When MLE has no closed-form solution (often), we use gradient-based methods, e.g. gradient descent (GD):

Algorithm 1 Gradient Descent

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

- Given previous questions, $p_{\boldsymbol{w}}$ is logistic regression, J is answer to question (e), g is Eq. 29.

Training Log-linear Classifiers

Question g) (2)

- 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.

Training Log-linear Classifiers

Answer g) (1)

- **SGD**:

Algorithm 2 Stochastic Gradient Descent

```
 1: procedure GRADIENT DESCENT(Dataset $X$, Objective function $J$, Model $p_{\boldsymbol{w}}$, Learning rate $lr$, Number of Epochs $N$)
 2:     $\boldsymbol{w} \leftarrow random()$        // random weight initialization
 3:     for $i \in \{1, \dots N\}$ do        // we iterate over epochs
 4:         for $x_i \in X$ do        // we iterate over the dataset
 5:             $g_i \leftarrow \nabla_{\boldsymbol{w}} J(p_{\boldsymbol{w}}(x_i))$
 6:             $\boldsymbol{w}_{i+1} \leftarrow \boldsymbol{w}_i - lr \cdot g_i$        // update rule
 7:         end for
 8:     end for
 9:     Return $p_{\boldsymbol{w}}$
10: end procedure
```

Training Log-linear Classifiers

Answer g) (2)

- **BGD**:

Algorithm 3 Batch Gradient Descent

```
1: procedure GRADIENT DESCENT(Dataset $X$, Objective function $J$, Model $p_{\boldsymbol{w}}$, Learning rate $lr$, Number of Epochs $N$, Batch Size $b$)
2:     $\boldsymbol{w} \leftarrow random()$        // random weight initialization
3:     for $i \in \{1, \dots N\}$ do
4:         for $batch = [x_1, \dots, x_b]$ where $batch$ is sampled without replacement from $X$ do
5:             $g_i \leftarrow \nabla_{\boldsymbol{w}} J(p_{\boldsymbol{w}}(batch))$
6:             $\boldsymbol{w}_{i+1} \leftarrow \boldsymbol{w}_i - lr \cdot g_i$        // update rule
7:         end for
8:     end for
9:     Return $p_{\boldsymbol{w}}$
10: end procedure
```

Training Log-linear Classifiers

Answer g) (3)

- Note that we still rely on gradients that need to be computed every time we update our parameters.
- But we don't compute gradients manually when working with these models.
- This is generally handled by tools like Autograd.
- But it's important to understand the process underlying the training of models, despite it being automatic, so we can reason about model performance as a function of its training process.

Training Log-linear Classifiers

Question 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?

Training Log-linear Classifiers

Answer h) (1)

- **BGD**:

Algorithm 4 Batch Gradient Descent With Shuffling

```
1: procedure GRADIENT DESCENT(Dataset $X$, Objective function $J$, Model $p_{\boldsymbol{w}}$, Learning rate $lr$, Number of Epochs $N$, Batch Size $b$)
2:     $\boldsymbol{w} \leftarrow random()$        // random weight initialization
3:     for $i \in \{1, \dots N\}$ do
4:         for $batch = [x_1, \dots, x_b]$ where $batch$ is sampled without replacement from $X$ do
5:             $g_i \leftarrow \nabla_{\boldsymbol{w}} J(p_{\boldsymbol{w}}(batch))$
6:             $\boldsymbol{w}_{i+1} \leftarrow \boldsymbol{w}_i - lr \cdot g_i$        // update rule
7:         end for
8:         shuffle($X$)
9:     end for
10:    Return $p_{\boldsymbol{w}}$
11: end procedure
```

Training Log-linear Classifiers

Answer h) (2)

- While GD considers the entire dataset when computing gradients, and SGD treats all examples equally, BGD uses a subset of the dataset to compute each gradient.
- Shuffling the dataset at the end of each epoch is a common practice to avoid using the same subset of examples for each BGD update, as this may bias the training process in favor of some examples over others.
- BGD is by far the most method used in NLP training.
- It has been empirically shown that SGD and BGD can converger faster than GD, but they can also be more unstable.