Methods

#### Topics

1. Multiclass classification
2. One-class classification
3. Multi-Label Classification
4. Ensemble Learning

* Boosting and Bagging
  + Random Forest
  + Gradient Boosting

1. Learning to Label Sequences
2. Sequence-to-Sequence Learning
3. Active Learning
4. Semi-Supervised Learning
5. One-Shot Learning
6. Zero-Shot Learning

# 1. Multiclass Classification

For multiclass classification problems, we extend the logistic regression to **softmax regression algorithm.** This is done by replacing sigmoid function in logistic regression with softmax function.

The loss function in softmax regression is typically the cross-entropy loss (also known as log-loss). Given a set of training examples, the cross-entropy loss measures the dissimilarity between the predicted probabilities and the true class labels.

- log a1, if y == 1

- log a2, if y == 2

L = :

:

- log an, if y == n

Where aj = ezj / ΣNk=1 = P(y = j | x) , where N is the number of classes

# 2. One class classification

One-class classification, also known as unary classification or class modeling, tries to identify objects of a specific class among all objects, by learning from a training set containing only the objects of that class. The most widely used in practice are one-class Gaussian, one-class k-means, one-class kNN, and one-class SVM.

# 3. Multi-Label Classification

In some situations, more than one label is appropriate to describe an example from the dataset. In this case, we talk about the multi-label classification. The only difference with the usual multiclass problem is that now we have a new hyperparameter: **threshold**. If the prediction score for some label is above the threshold, this label is predicted for the input feature vector. In this scenario, multiple labels can be predicted for one feature vector.

# 4. Ensemble learning

Ensemble learning is a learning paradigm that, instead of trying to learn one super-accurate model, focuses on training a large number of low-accuracy models and then combining the predictions given by those weak models to obtain a high-accuracy **meta-model.**

Low-accuracy models are usually learned by weak learners, that is, learning algorithms that cannot learn complex models, and thus are typically fast at the training and at the prediction time. The most frequently used weak learner is a decision tree learning algorithm in which we often stop splitting the training set after just a few iterations. The obtained trees are shallow and not particularly accurate, but the idea behind ensemble learning is that if the trees are not identical and each tree is at least slightly better than random guessing, then we can obtain high accuracy by combining a large number of such trees. To obtain the prediction for input x, the predictions of each weak model are combined using some sort of **weighted voting**. Two principal ensemble learning methods are boosting and bagging.

#### Boosting and Bagging

**Boosting** consists of using the original training data and iteratively creating multiple models by using a weak learner. Each new model would be different from the previous ones in the sense that the weak learner, by building each new model tries to “fix” the errors which previous models make. The final ensemble model is a certain combination of those multiple weak models built iteratively.

**Bagging** consists of creating many “copies” of the training data (each copy is slightly different from another) and then apply the weak learner to each copy to obtain multiple weak models and then combine them. A widely used and effective machine learning algorithm based on the idea of bagging is random forest.

#### Random Forest

Random Forest is a **specific implementation of bagging that is tailored for decision trees** and **incorporates feature randomness** to further enhance its performance and robustness. In a Random Forest, multiple decision trees are trained on different subsets of the data using sampling with replacement and with random feature subsets. The predictions from these individual trees are then combined to make a final prediction, typically through a majority vote for classification tasks or averaging for regression tasks.

Random forest is one of the most widely used ensemble learning algorithms. The reason is that by using multiple samples of the original dataset, we reduce the variance of the final model, lowering overfitting.

#### Gradient Boosting

Gradient boosting is a modified boosting algorithm that implements a gradient descent (in a non traditional way) to calculate the error caused by the ensemble trees. This error is used to update the trees, training the model and reducing the error. However, instead of getting the gradient directly, we use its proxy in the form of residuals: they show us how the model has to be adjusted so that the error (the residual) is reduced. It reduces underfitting, but can lead to overfitting. So careful selection of hyperparameters is adviced.

Gradient boosting is one of the most powerful machine learning algorithms—not just because it creates very accurate models, but also because it is capable of handling huge datasets with millions of examples and features. It usually outperforms random forest in accuracy but, because of its sequential nature, can be significantly slower in training.

# 5. Sequence-to-Sequence Learning

**Sequence Labeling** is the problem of labelling a sequence of data. For eg, classifying each word in a string to {‘noun’, ‘verb’, ‘adjective’}.

**Sequence-to-Sequence learning (seq2seq learning)** is a generalisation of the sequence labeling problem. The crux of the problem is to take in a sequence and output another sequence (of different length) according to the situation. Eg: text generation, machine translation, etc.

**Word embeddings**

Word embeddings are a type of representation of words as numerical values in a way that allows machines to understand and process the meaning of words in a meaningful and context-aware manner. This can be done by training a neural network on a lot of data, and asking the nn to generate an n dimensional vector to represent a word. This also leads to simal words having similar embeddings. Popular word embedding techniques include Word2Vec, GloVe, and FastText.

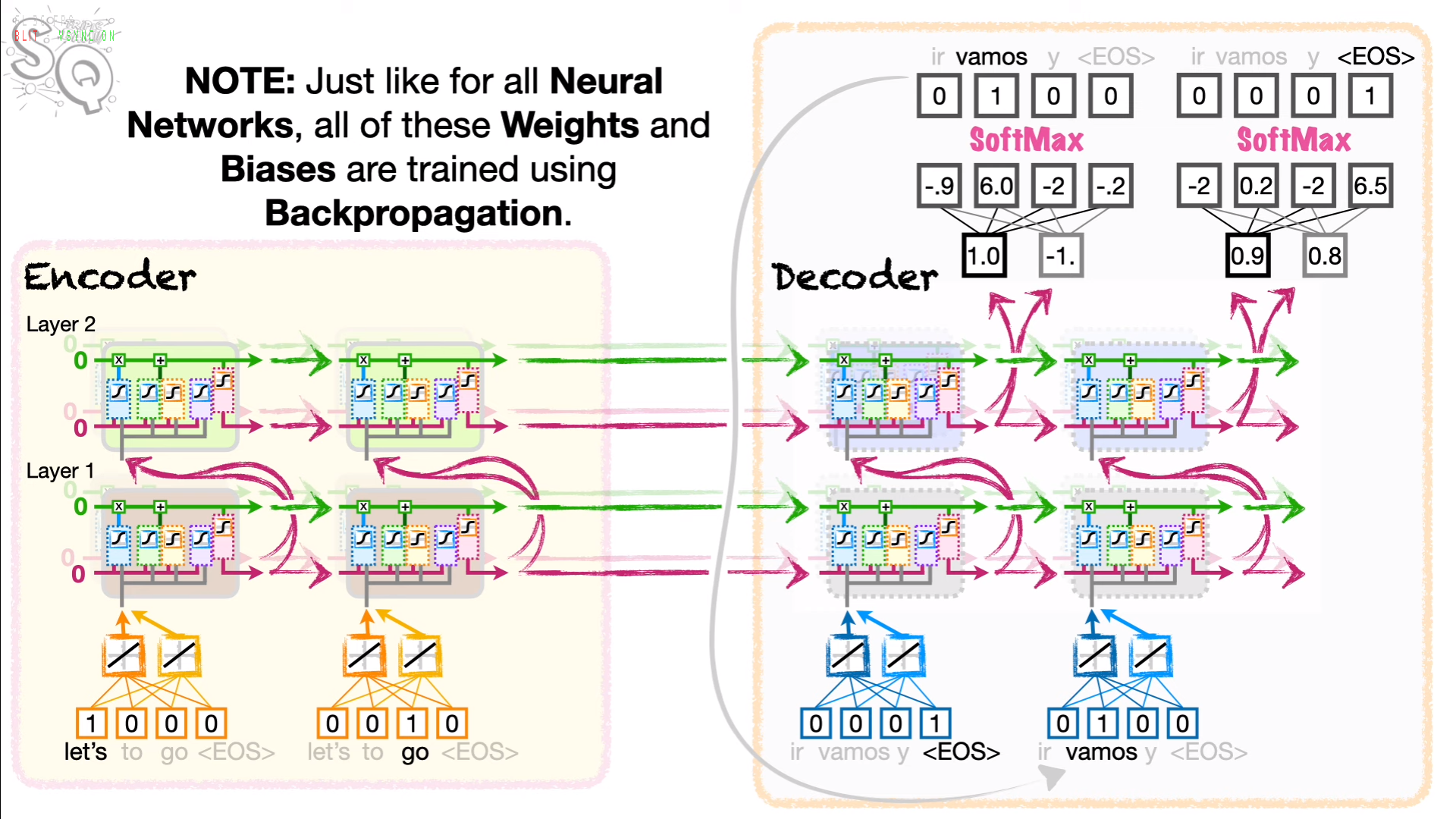
**Word2vec** does word embeddings, but also does two more things

1. Continuous bag of words: It looks at surrounding words to create the embedding.
2. Skip Gram: Uses words in middle to predict surrounding words.

The network architecture used to solve seq2seq problems have two parts: an encoder and a decoder.

**Encoder:** We input the embedding of the data into the encoder, which is converted into a context vector by the encoder layer. The embeddings provide a static representation of each word and are usually pre-trained on large collection of texts to capture word semantics but do not capture the context of the entire sequence. The encoder's role is to process these embeddings and produce a context vector that summarizes the sequence's context.

**Decoder:** The context vector from the encoder is used to initialise the decoder. The role of the whole decoder setup is to decode the context vector into the output sentence. It has a separate embeding layer that describes its vocabulory. The input into this layer (for inputing the output of the previous timestamp) is passed throught this embedding layer. The output from the decoder layer is passed to a fully connected layer, and into a softmax function, which gives out the probability of each word (in decoder vocabulory) as output.



Traditionally, sequence-to-sequence (seq2seq) models used recurrent neural networks (RNNs), including Long Short-Term Memory (LSTM) networks, for both encoders and decoders. LSTMs are capable of handling sequential data and have been widely used in NLP tasks. However, more modern seq2seq models, especially those based on the Transformer architecture, have become popular and do not rely on LSTMs.

Attention

**Learn later deeply**

Active Learning

The idea is to start training a supervised model with relatively few labeled examples, and a large number of unlabeled ones, and then label only those examples that contribute the most to the model quality. It is usually applied when obtaining labeled examples is costly.

Data density and uncertainty based model: For each unlabeled example x, the following importance score is computed: density(x) \* uncertainty(x). Density for the example x can be obtained by taking the average of the distance from x to each of its k nearest neighbors. Once we know the importance score of each unlabeled example, we pick the one with the highest importance score and ask the expert to annotate it.

SVM based model: The support vector-based active learning strategy consists in building an SVM model using the labeled data. We then ask our expert to annotate the unlabeled example that lies the closest to the hyperplane that separates the two classes.

Semi-Supervised Learning

In semi-supervised learning (SSL) we also have labeled a small fraction of the dataset; most of the remaining examples are unlabeled. Our goal is to leverage a large number of unlabeled examples to improve the model performance without asking for additional labeled examples.

self-learning

one frequently cited SSL method is called self-learning. In self-learning, we use a learning algorithm to build the initial model using the labeled examples. Then we apply the model to all unlabeled examples and label them using the model. If the confidence score of prediction for some unlabeled example x is higher than some threshold (chosen experimentally), then we add this labeled example to our training set, retrain the model and continue like this until a stopping criterion is satisfied. We could stop, for example, if the accuracy of the model has not been improved during the last m iterations.