Document covers the following:

1. Input encoding for NLP tasks
2. Feed-forward networks
3. Convolutional Neural networks(CNN)
4. Recurrent neural networks (RNN)
5. Recursive neural networks
6. Computation graph abstraction for automatic gradient computation

* For deep learning reference, this [deepLearningBook](http://www.deeplearningbook.org/) is recommended.

Introduction

* Feature - concrete linguistic input such as word, suffix or POS tag
* input vector - actual input to the neural network classifier
* input vector entry - an instance of the input vector
* Matrices are represented using upper case and vectors with lower case
* All vectors are row vectors

Neural Network Architectures

2 kinds of architectures are discussed.

1. Feed forward networks
2. Recurrent/Recursive networks

Feed forward networks

* Multi layer perceptrons and CNNs (Networks with convolutional and pooling layers)
* Ability to include pre-trained word embeddings
* Applications of feed-feed forward neural networks
  + CCG super tagging
  + dialog state tracking
  + pre-ordering for statistical machine translation
  + language modelling
  + sentiment classification
  + factoid question answering
* Applications of CNNs
  + document classification
  + short text categorization
  + sentiment classification
  + relation type classification between entities
  + event detection
  + paraphrase identification
  + semantic role labeling
  + question answering
  + predicting box office revenue based on critics’ reviews
  + modelling text interestingness
  + modelling releation between character-sequences and POS tags

Recurrent models

* Applications
  + language modelling
  + sequence tagging
  + machine translation
  + depedency parsing
  + sentinent analysis
  + noisy text normalization
  + dialog state tracking
  + response generation
  + modelling releation between character-sequences and POS tags

Recursive models

* Applications
  + constituency and dependency parse-reranking
  + discouse parking
  + semantic relation classigication
  + political ideology detection based on parsed trees
  + sentiment classification
  + target-dependent sentiment classification
  + question answering

Feature representation

The biggest jump when moving from sparse-input linear models to neural-network based models is to stop representing them instead as dense vectors. Each core feature is embedded into a d dimensional space and represented as a vector in that space. These embedded vectors can be trained like any other parameter of the neural network.

Using sparse, one-hot vectors when traning NN amounts to dedicating the first layer to learn dense embeddings for each feature vector.

General structure of NLP classification system:

1. Extract a set of core linguistic features relevant for predicting output class
2. Transform the core features into vectors (dense vector representations)
3. Combine the vectors
4. Feed the combined vectors into non-linear classifier(feed-foward NN)

Advantages of using dense representations:

* Generalization. Dense vector representation captures simiarlity between simiar vectors.
* Computational. Dense vectors need less compute power.

Common Non-linearities or activation functions

1. Sigmoid: S shaped function σ(x) = 1/(1 + e−x) transforming each value in the range [0,1]
2. Hyperbolic tangent or tanh: S shaped function transoforming values in range [-1,1]. tanh(x) = e2x−1/e2x +1
3. Hard tanh: approximation of tanh,faster to compute and take derivatives of.
   1. hardtanh(x) = - 1 if x < -1; 1 if x >1; x otherwise
4. Rectifier(ReLU): ReLU(x) = max(0,x)
5. Softmax: typically used for output layers. Produces probability distribution over output classes.
6. Others: cube and tanh cube activation functions
7. ReLU works better than tanh and tanh works better than sigmoid

Loss functions

Loss is a scalar quantity measuring for network output yhat given the true output y. Loss can be any arbitrary function that maps 2 vectors to a scalar. For practical purposes, a loss function for which gradient or derivative is easily calculated is preferred. Some of the common loss functions:

* Hinge loss for binary classification
* Hinge loss for multi class classification
* Log loss
* Categorical cross entropy loss or cross entropy loss
* Ranking loss

Word Embeddings

Representing each feature as vector in a low dimensional space.Common approaches to create or generate word embeddings.

* Random initialization: With supervised training data, initialize the feature embeddings with radom values and tune them with network learning
* Supervised Task specific Pre-training: Use pre-trained vectors on an auxilary task and re-train them on the current task to fine tune the vectors
* Unsupervised Pre-training: The techniques to learn word vectors here are essentially same as supervised techniques. Instead of supervison for the task at hand, we create large number of supervised tasks from the raw text. At the end, word embeddings generated by these supervised tasks are usefull for the unsupervised task.
  + word2vec
  + GLoVE
  + Collobert and Weston embeddings algorithm
* Training objectives: Given word ‘w’ and context ‘c’, auxilary tasks are used to learn the word embeddings. Like predict the word ‘w’ given context ‘c’.
* Choice of context (used to generate word embeddings):
  + Window approach
    - CBOW
    - Skip-gram
  + Window size
    - Large windows - learn topical similarities. like dog, leash and bark are grouped together
    - Small windows - learn functional and syntactic similarities. like ‘poodle’, ‘pitbull’, ‘jack russel’ are grouped together
  + Positional windows: Using positional context, factoring in where in context a context word occurs, along with smaller window contexts produce similarities that are more syntactic, with a strong tendency of grouping together words that share a part of speech, as well as being functionally similar in terms of their semantics.
* Syntactic window: In some work, instead of using linear context with in a sentense, the text is parsed by a dependency parser and parse tree is contructed(?). Then context of the word is to be take from its close proximity in the parse tree.
* Character-based and sub-word representations: Deriving the word vectors from the characters that constitute it. And also using sub-words that are part of main word.

Neural Network Training

Stochastic Gradient Descent: Calculate error on each training example, compute the gradient of the error. And adjust the NN parameters in the direction of gradient @learning rate. Repeat untill the loss or error is acceptable.

Minibatch SGD: Calculating error on each training example may result in noise which inturn results in inaccurate gradients. An effective way to avoid is to caluclate and estimate erorr & gradient based on set of m samples. Advantages:

* Large m provide better corpus wide gradients
* Small m allow frequent updates and faster convergence
* improved training efficiency by running mulitple minibatch trainings in parallel(GPUs)

SGD+ momentum and Nesterov momentum: are variants are SGD in which previous gradients are accumulated and affect the currnet udpdate.

AdaGrad, AdaDelta, RMSProp, Adam: adaptive learning rate algorithms designed to select the learning rate for each minibatch potentially removing the need of learning rate scheduling.

The computational Graph Abstraction - to be studied later.

Optimization Issues

* Initialization: When network parameters are initialized at random values, there is a risk of getting stuck at local minima. In order to avoid this, multiple starts starting from different initial points is recommended. Also initialization of the parameters has important effect on training success. 2 kinds of initializations used are xavier initialization and sampling from zero-mean gaussian distribution. The later initialization works better for image classification using deep learning.
* Vanishing and Exploding gradients:
  + In deep networks, the gradients can be lost as they propagate back through the network. The deeper the network is the severe the problem. This can be combated with shallow networks, stepwise training. Also specialized network architetures like LSTM and GRU are used.
  + To deal with exploding gradients, clipping is performed. Gradients are clipped if their norms exceed certain threshold.
* Saturated and Deal Neurons:
  + Layers with tanh and sigmoid activations can be saturated meaning their outputs are all close to one.
  + Layers with ReLU activations can die meaning their most or all of their outputs are negative and clipped at 0.
* Shuffling: Order of the tranining examples presented to the network is important. It is advised to shuffle the training example before each pass.
* Learning rate: Large learning rate will prevent the network to converge on the effective solution; very small learning rates are computational expensive and take very long time to converge. Using learning rate scheduling helps to change learning rate as the training proceeds.
* Minibatches:Some problems benefit from training with large minibatch sizes.

Regularization

Overfitting can be avoided to certain extent by regularization. L2 regularization method is commonly used that places penalty on parameters with large values. Another method, dropout works by dropping(setting weights = 0) half of the neurons in the network in a specific layer in each training example. Dropout method is designed to prevent the network from learning to rely on specific weights. Dropout is effective in image classifications and NLP applications of neural networks.