Deep Learning for Natural Language Processing
Text Classification

Karl Moritz Hermann

kmh@google.com

7 Feb 2017

Overview

This lecture discusses text classification.
As part of that we'll discuss the following

- Generative and discriminative models
- Naïve Bayes
- Logistic regression
- Text representation (BOW, Features, RNNs, Convolutions)
- Softmax Classifiers
- Practical Aspects of Classifier Training

Important Message

Good Day,

My name is Dr William Monroe, a staff in the Private Clients Section of a well-known bank, here in London, England. One of our accounts, with holding balance of £15,000,000 has been dormant and last operated three years ago. From my investigations, the owner of the said account, John Shumejda died on the 4th of January 2002 in a plane crash.

I have decided to find a reliable foreign partner to deal with. I therefore propose to do business with you, standing in as the next of kin of these funds from the deceased.

This transaction is totally free of risk and troubles as the fund is legitimate and does not originate from drug, money laundry or terrorism.

On your interest, let me hear from you URGENTLY.

Best Regards,
Dr William Monroe Financial Analysis and Remittance Manager

Why Classify?

Classification Tasks

- Is this e-mail spam?
- Positive or negative review?
- What is the topic of this article?
- Predict hashtags for a tweet
- Age/gender identification
- Language identification
- Sentiment analysis
- ...

Types of Classification Tasks

- Binary classification (true, false)
- Multi-class classification (politics, sports, gossip)
- Multi-label classification (#party #FRIDAY #fail)
- Clustering (labels unknown)

Classification Methods

1. **By hand**
2. **Rule-based**
3. **Statistical**

Classification Methods

1. **By hand**
   E.g. Yahoo in the old days
   ✔ Very accurate and consistent assuming experts
   ✘ Super slow, expensive, does not scale
2. **Rule-based**
   E.g. Advanced search criteria ("site:ox.ac.uk")
   ✔ Accuracy high if rule is suitable
   ✘ Need to manually build and maintain rule-based system.
3. **Statistical**
   This lecture
   ✔ Scales well, can be very accurate, automatic
   ✘ Requires classified training data. Sometimes a lot!

Statistical Text Classification

Assume some text represented by d and some class c. We want to learn the probability of d being of class c:

$$P(c|d)$$

Key questions:

- How to represent d.
- How to calculate $P(c|d)$.

Text Classification in Two Parts

Think of text classification as a two stage process:

Representation

Process text into some (fixed) representation.

How to learn d

Classification

Classify document given that representation.

How to learn $P(c|d)$

Possible Representations for Text

- **Bag of Words (BOW)**
  - Easy, no effort required.
  - Variable size, ignores sentential structure.
- **Hand-crafted features**
  - Full control, can use of NLP pipeline, class-specific features
  - Over-specific, incomplete, makes use of NLP pipeline.
- **Learned feature representation**
  - Can learn to contain all relevant information.
  - Needs to be learned.

Generative vs. Discriminative Models

Generative (joint) models $P(c, d)$

- Model the distribution of individual classes and place probabilities over both observed data and hidden variables (such as labels)
- E.g. n-gram models, hidden Markov models, probabilistic context-free grammars, IBM machine translation models, Naïve Bayes, ...

Discriminative (conditional) models $P(c|d)$

- Learn boundaries between classes. Take data as given and put probability over the hidden structure given the data.
- E.g. logistic regression, maximum entropy models, conditional random fields, support-vector machines, ...

Naïve Bayes classifier (1/4)

Bayes' Rule:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}$$

We can simplify this in a classification scenario and ignore the denominator $P(d)$ because it is independent of class choice c:

$$\begin{aligned} P(c|d) &\propto P(c)P(d|c) \\ &\propto P(c) \prod_{1 \leq i \leq n_d} P(t_i|c) \end{aligned}$$

This estimates the probability of document d being in class c, assuming document length n_d and tokens t.

Naïve Bayes classifier (2/4)

$$P(c|d) \propto P(c) \prod_{1 \leq i \leq n_d} P(t_i|c)$$

$P(c)$ and $P(t|c)$ can be estimated from labelled training data:

$$P(c) = \frac{D_c}{D}$$

$$P(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

Independence Assumptions

Note that we assume $P(t_i|c) = P(t_j|c)$ independent of token position. This is the **naïve** part of Naïve Bayes.

Naïve Bayes classifier (3/4)

The *best* class is the maximum a posteriori (MAP) class:

$$c_{map} = \operatorname*{argmax}_{c \in C} P(c|d) = \operatorname*{argmax}_{c \in C} P(c) \prod_{1 \leq i \leq n_d} P(t_i|c)$$

Multiplying tons of small probabilities is tricky, so log space it:

$$c_{map} = \operatorname*{argmax}_{c \in C} \left(\log P(c) + \sum_{1 \leq i \leq n_d} \log P(t_i|c) \right)$$

Finally: zero probabilities are bad. Add smoothing:

$$P(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}} \quad \Rightarrow \quad P(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} T_{ct'} + |V|}$$

This is Laplace or add-1 smoothing. There are many alternatives.

Naïve Bayes classifier (4/4)

Advantages

- Simple
- Interpretable
- Fast (linear in size of training set and test document)
- Text representation trivial (bag of words)

Drawbacks

- Independence assumptions often too strong
- Sentence/document structure not taken into account
- Naïve classifier has zero probabilities; smoothing is awkward

Quiz

Question

Is Naïve Bayes a generative or a discriminative model?

Quiz

Question

Is Naïve Bayes a generative or a discriminative model?

Naïve Bayes is a **generative** model!

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

$$P(c|d)P(d) = P(d|c)P(c)$$

$$= \mathbf{P(d, c)}$$

- While we use a conditional probability $P(c|d)$ for classification, we model the joint probability of c and d.
- This means it is trivial to invert the process and generate new text given a class label.

Feature Representations

A feature representation (of text) can be viewed as a vector where each element indicates the presence or absence of a given feature in a document.
Note: features can be binary (presence/absence), multinomial (count) or continuous (eg. TF-IDF weighted).

Feature	Coefficient	Weight
bias	β_0	0.2
"prozac"	β_1	1.4
"school"	β_2	-0.4
"Dear Friend"	β_3	1.8
"nigeria"	β_4	2.0
"homework"	β_5	-1.7

What does this feature representation classify?

Logistic Regression (1/7)

If we only want to classify text, we do not need the full power of a generative model, but a discriminative model is sufficient.

We only want to learn $P(c|d)$.

A general framework for this is logistic regression.

logistic	because is uses a logistic function
regression	combines a feature vector (d) with weights (β) to compute an answer

Logistic Regression (2/7)

Binary case:

$$P(\text{true}|d) = \frac{1}{1 + exp(\beta_0 + \sum_i \beta_i X_i)}$$

$$P(\text{false}|d) = \frac{exp(\beta_0 + \sum_i \beta_i X_i)}{1 + exp(\beta_0 + \sum_i \beta_i X_i)}$$

Multinomial case:

$$P(c|d) = \frac{exp(\beta_{c,0} + \sum_i \beta_{c,i} X_i)}{\sum_{c'} exp(\beta_{c',0} + \sum_i \beta_{c',i} X_i)}$$

where X are the features contained in d.

Logistic Regression (3/7)

The binary and general functions*(*)* for the logistic regression

$$P(\text{true}|d) = \frac{1}{1 + exp(\sum_i \beta_i X_i)} \quad P(c|d) = \frac{exp(\sum_i \beta_{c,i} X_i)}{\sum_{c'} exp(\sum_i \beta_{c',i} X_i)}$$

can be simplified as follows

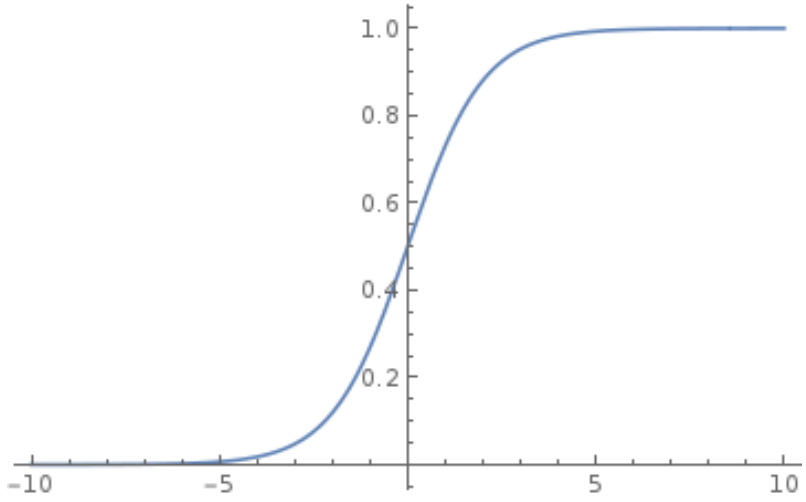
$$P(c|d) = \frac{1}{1 + exp(-z)} \qquad P(c|d) = \frac{exp(z_c)}{\sum_{c'} exp(z_{c'})}$$

which are referred to as the logistic and softmax function.

** This is not the most general form for the logistic regression, but a suitable specification for the purposes of this course.*

Logistic Regression (4/7)

Logistic Function



Softmax Function

- Multinomial generalisation of logistic function
- Takes output of K distinct linear functions and returns a probability distribution over those outputs

Logistic Regression (5/7)

- Given this model formulation, we want to learn parameters β that maximise the conditional likelihood of the data according to the model.
- Due to the softmax function we not only construct a classifier, but learn probability distributions over classifications.
- There are many ways to chose weights β:

  Perceptron Find misclassified examples and move weights in the direction of their correct class

  Margin-Based Methods such as Support Vector Machines can be used for learning weights

  Logistic Regression Directly maximise the conditional log-likelihood via gradient descent (next slide).

Logistic Regression (6/7)

The conditional log-likelihood of the multinomial logistic regression is:

$$\log P(c|d,\beta) = \log \prod_{c,d\in(C,D)} P(c_n|d_n,\beta)$$

$$= \sum_{c,d\in(C,D)} \log P(c_n|d_n,\beta)$$

$$\cancel{\log P(c_n|d_n,\beta)} = \sum_{c,d\in(C,D)} \log \frac{exp(\sum_i \beta_{c,i} X_i)}{\sum_{c'} exp(\sum_i \beta_{c',i} X_i)}$$

Learning weights

✔ Derivative with respect to β is concave

✘ No closed-form solution

(Calculation left as excercise - or look up in the earlier lecture)

Logistic Regression (7/7)

Advantages

- Still reasonably simple
- Results are very interpretable
- Do not assume statistical independence between features!

Drawbacks

- Harder to learn than Naïve Bayes
- Manually designing features can be expensive
- Will not necessarily generalise well due to hand-crafted features

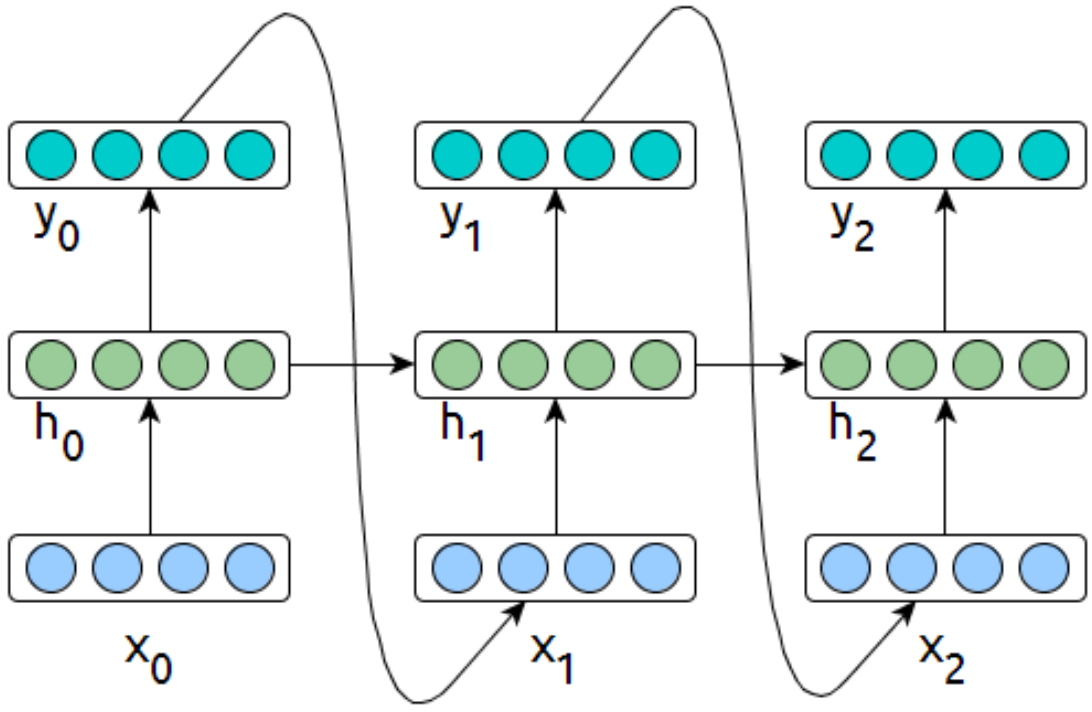
Break

This was the broad introduction to text classification

We will cover Deep Learning approaches next

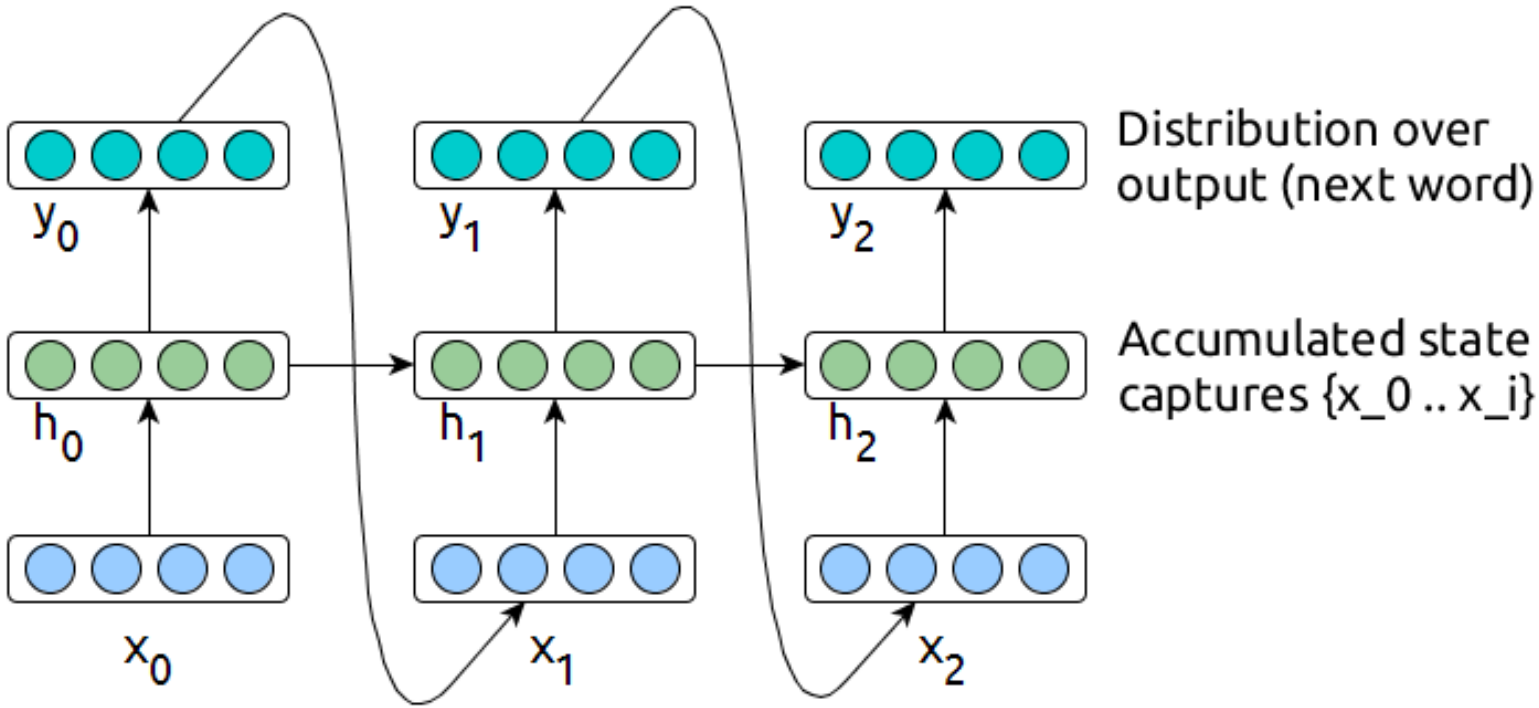
Recap: Recurrent Neural Networks

Recurrent Neural Network Language Model

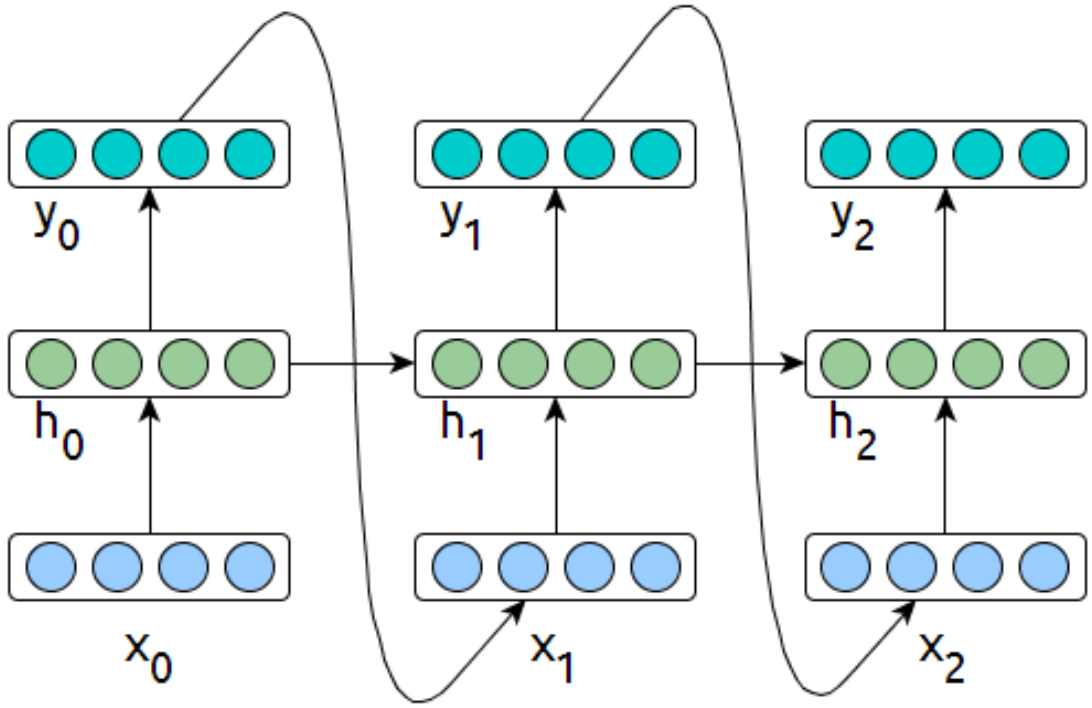


- Agnostic to actual recurrent function (LSTM, GRU, ..)
- Reads inputs x_i to accumulate state h_i and predict outputs y_i

Information contained in an RNN



Representing Text with an RNN



- h_i is a function of $x_{\{0:i\}}$ and $h_{\{0:i-1\}}$
- It contains information about all text read up to point i.
- The first half of this lecture was focused on learning a representation X for a given text
- h is precisely that

Logistic Regression Revisited

Effectively, $X = h_n$ where n is the length of a given input document.

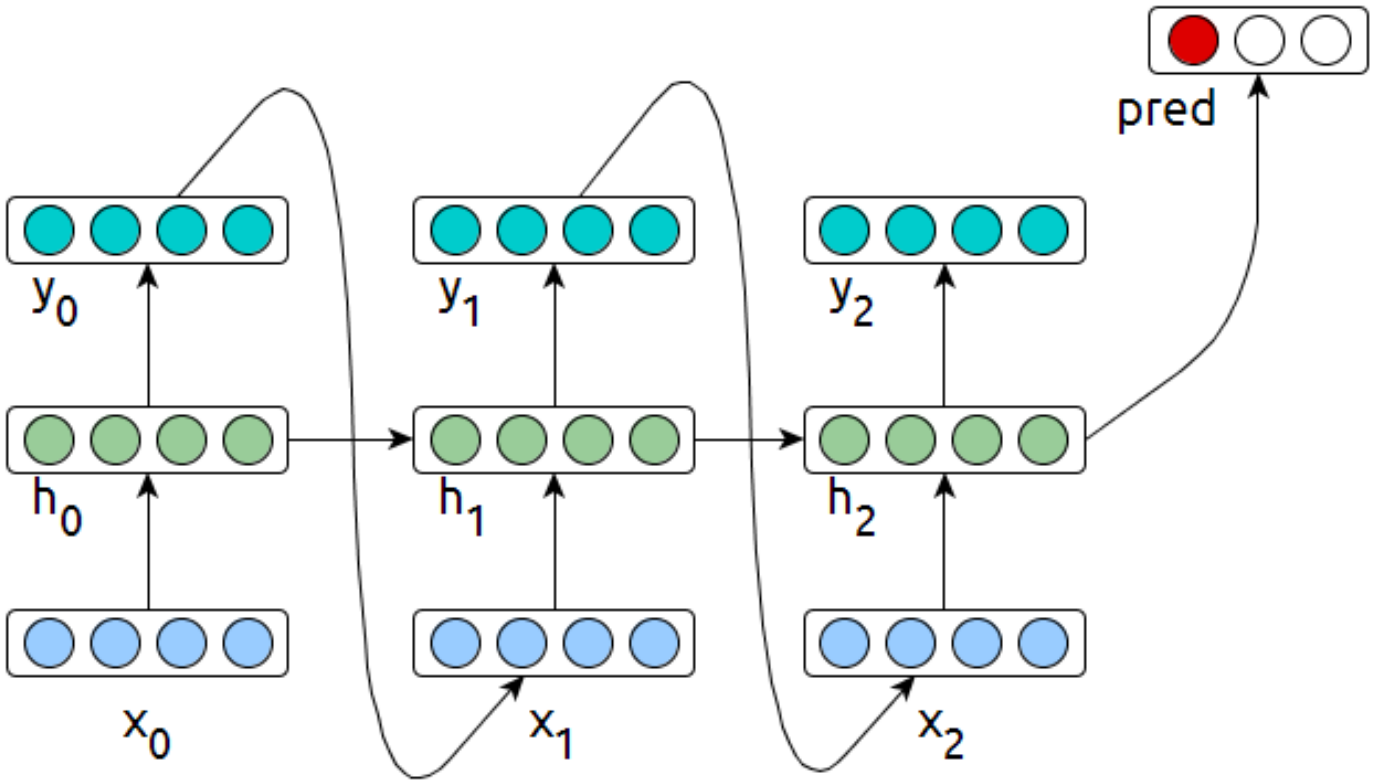
So in order to classify text we can simply take a trained language model (last week) and extract text representations from the final hidden state c_n.
Classification as before using a logistic regression:

$$P(c|d) = \frac{exp(\sum_i \beta_{c,i} h_{ni})}{\sum_{c'} exp(\sum_i \beta_{c',i} h_{ni})}$$

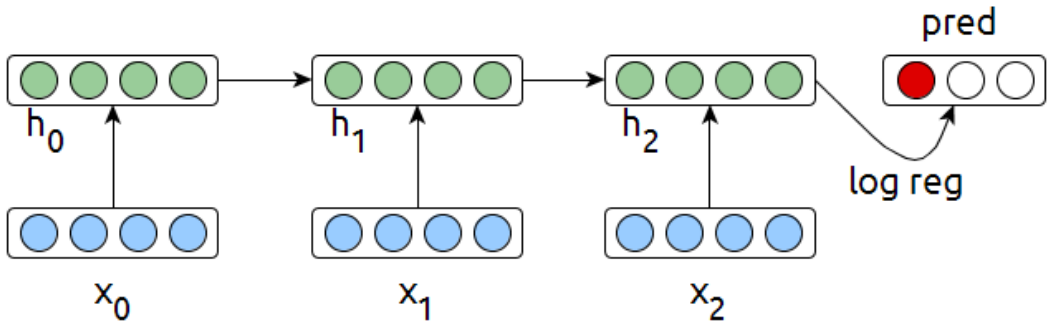
✔ Can use RNN + Logistic Regression *out of the box*
✔ Can in fact use any other classifier on top of h!
✘ How to ensure that h pays attention to relevant aspects of data?

Text Classification with an RNN

Move the classification function inside the network



Text Classification with an RNN



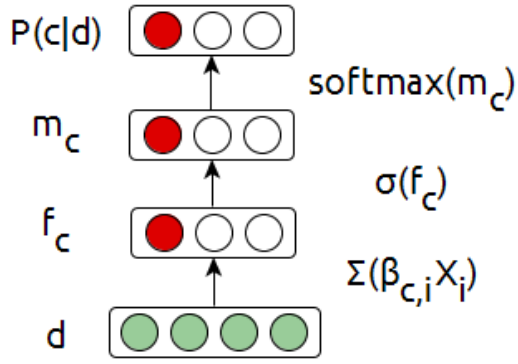
We do not need output layers y as we only care about $p(c|d)$!

Take RNN state as input $X = h_n$

Compute class weights $f_c = \sum_i \beta_{c,i} X_i$

Apply nonlinearity $m_c = \sigma(f_c)$

Apply softmax function $P(c|d) = \frac{exp(m_c)}{\sum_i exp(m_i)}$



Loss function for an RNN Classifier

You will have encountered this classifier already in the first lecture.
This is a simple Multilayer Perceptron (MLP).
We can train the model using the cross-entropy loss:

$$L_i = -\sum_c y_c \log P(c|d_i) = -\log\left(\frac{exp(m_c)}{\sum_j exp(m_j)}\right)$$

where $y_c = 1$ if $c = i, 0$ otherwise.

- Cross-entropy is designed to deal with errors on probabilities.
- Optimizing means minimizing the cross-entropy between the estiated class probabilities ($p(c|d)$ and the *true* distribution.
- There are many alternative losses (hinge-loss, square error, L1 loss) not discussed today.

Extension: Multi-Label Classification

What if a single data point may have multiple correct labels?

✘ The cross-entropy loss assumes there is only one correct label.

Option 1: Binary classifiers

For each possible label class (c) define a binary classifier (true/false) on class feature weights m_c. Assume $y_{ic} = 1$ if example i is in class c and 0 otherwise. The loss per example i is:

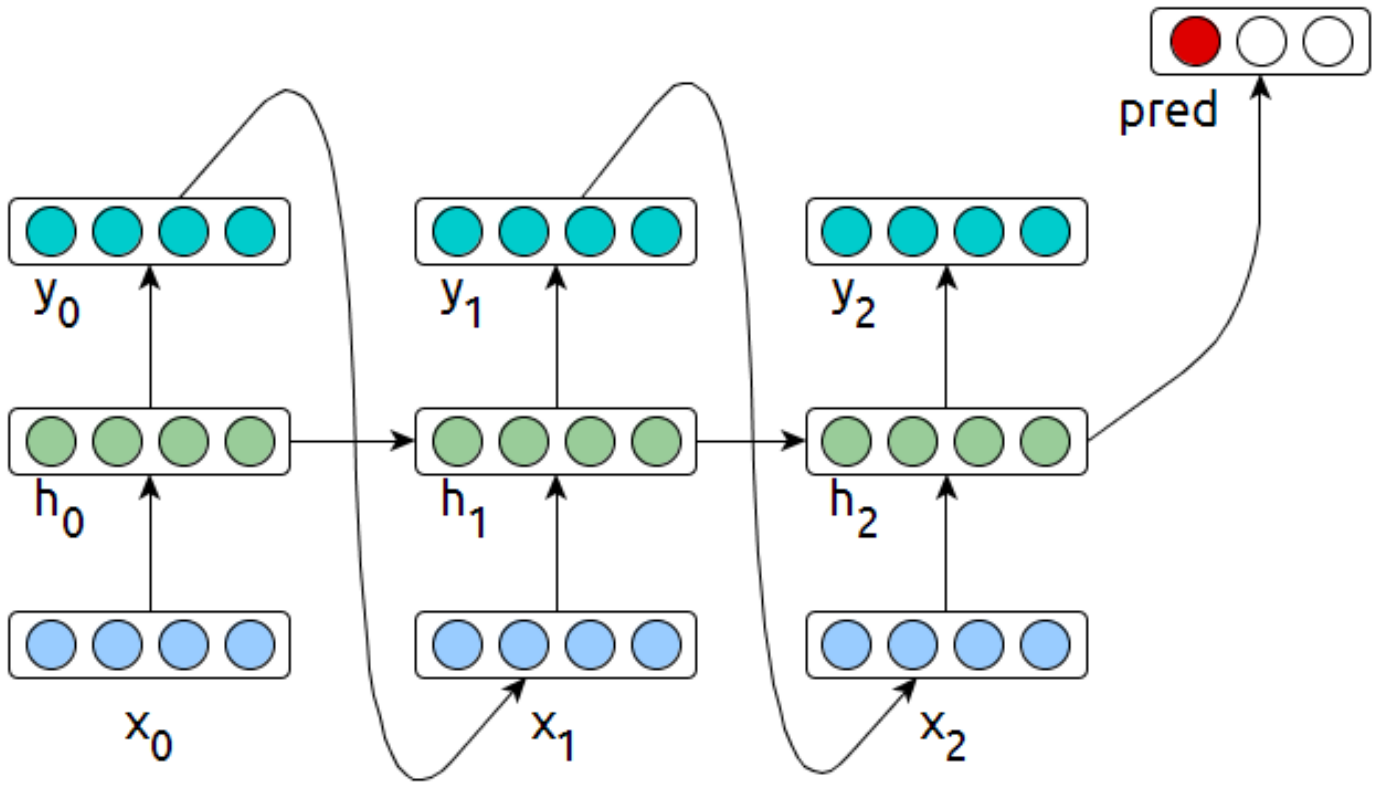
$$L_i = \sum_c y_{ic} \log(\sigma(f_c)) + (1 - y_{ic}) log(1 - \sigma(f_c))$$

$$\frac{\partial L_i}{\partial f_c} = y_{ic} - \sigma(f_c)$$

How else could you formulate a multi-label classifier?

Dual Objective RNN

In practice it may make sense to combine an LM objective with classifier training and to optimise the two losses jointly.

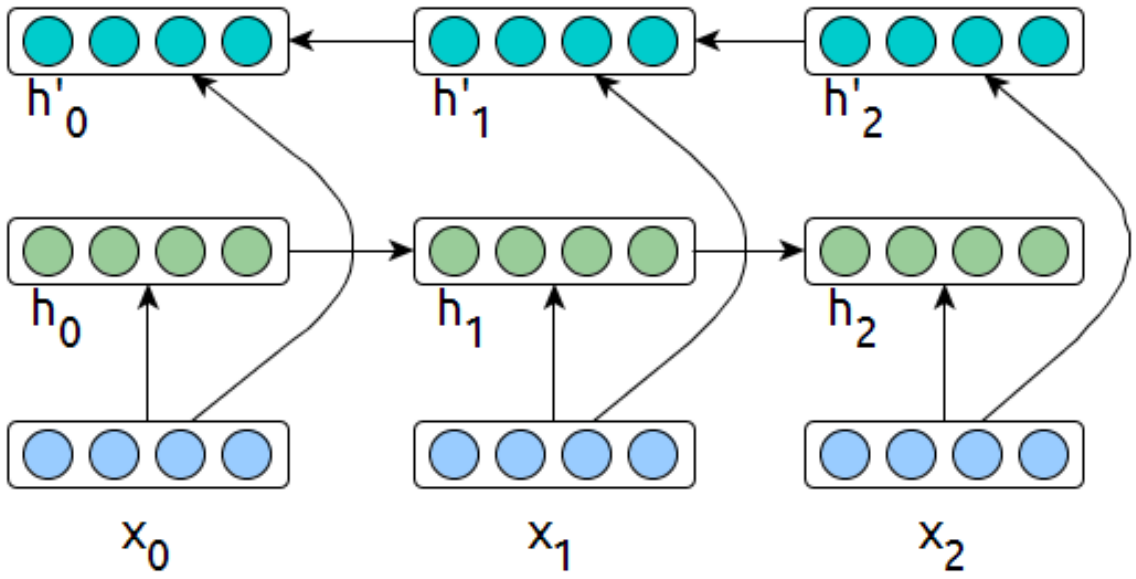


$$J = \alpha J_{\text{class}} + (1 - \alpha) J_{\text{lm}}$$

Such a joint loss enables making use of text beyond labelled data.

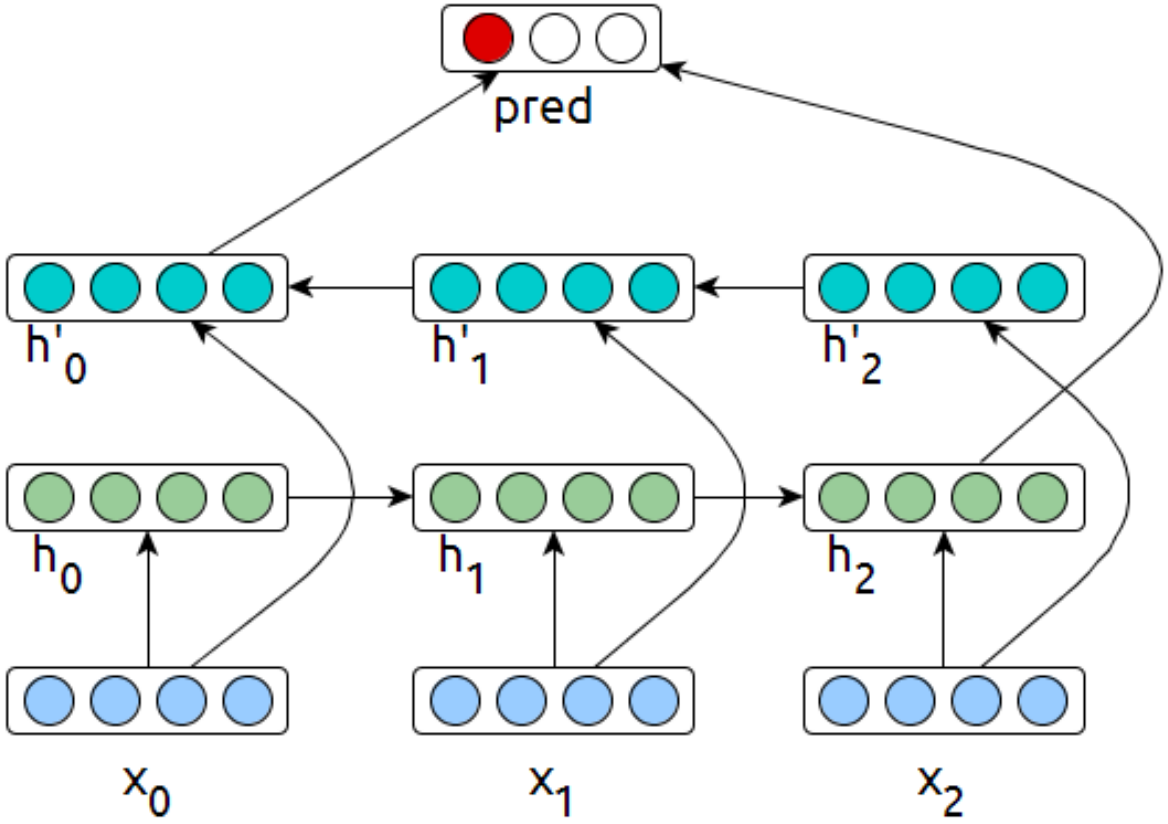
Bi-Directional RNNs

Another way to add signal is to process the input text both in a forward and in a backward sequence.



The update rules for this directly follow the regular forward-facing RNN architecture. In practice, bidirectional networks have shown to be more robust than unidirectional networks.

Bi-Directional RNNs



A bidirectional network can be used as a classifier simply by redefining d to be the concatenation of both final hidden states:

$$d = (\overrightarrow{h_n} \| \overleftarrow{h_0})$$

Quiz

Question

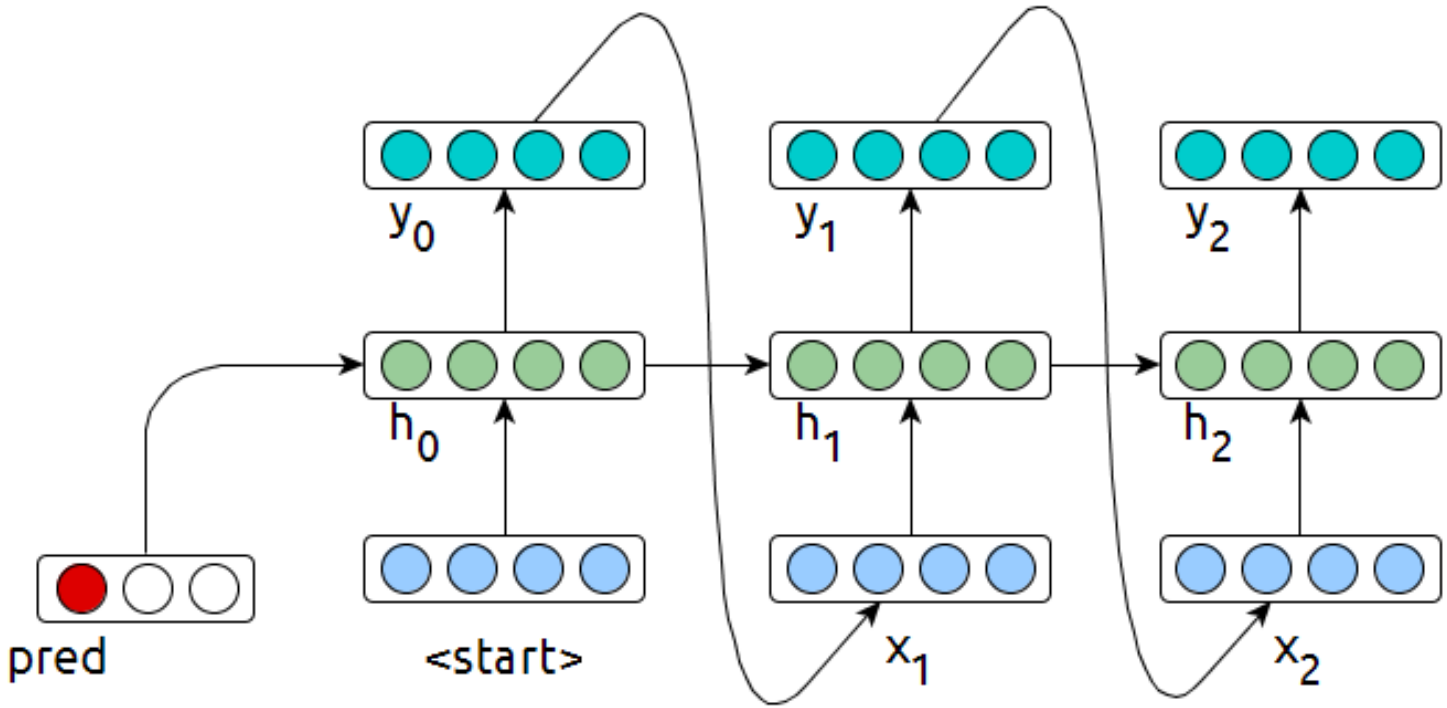
Is a simple RNN classifier a generative or discriminative model?

Quiz

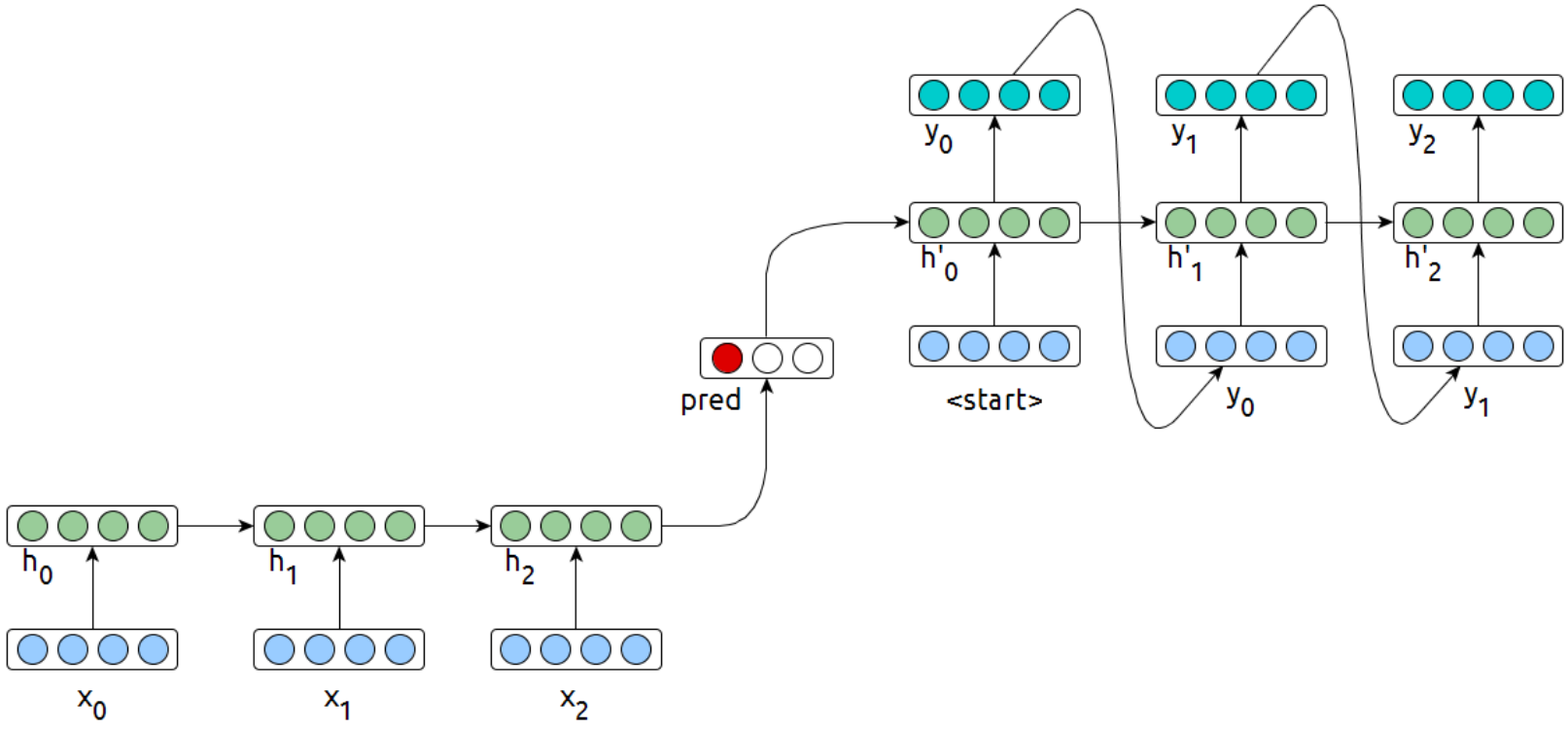
Question

Is a simple RNN classifier a generative or discriminative model?

RNN Classifiers can be either!



Generative Model



Encoder: discriminative (it does not model the probability of the text)

Joint-model: generative (learns both $P(c)$ and $P(d)$)

Non-Sequential Neural Networks

Outside of natural language a lot of data is not sequential. There are different architectures designed to deal with such data.

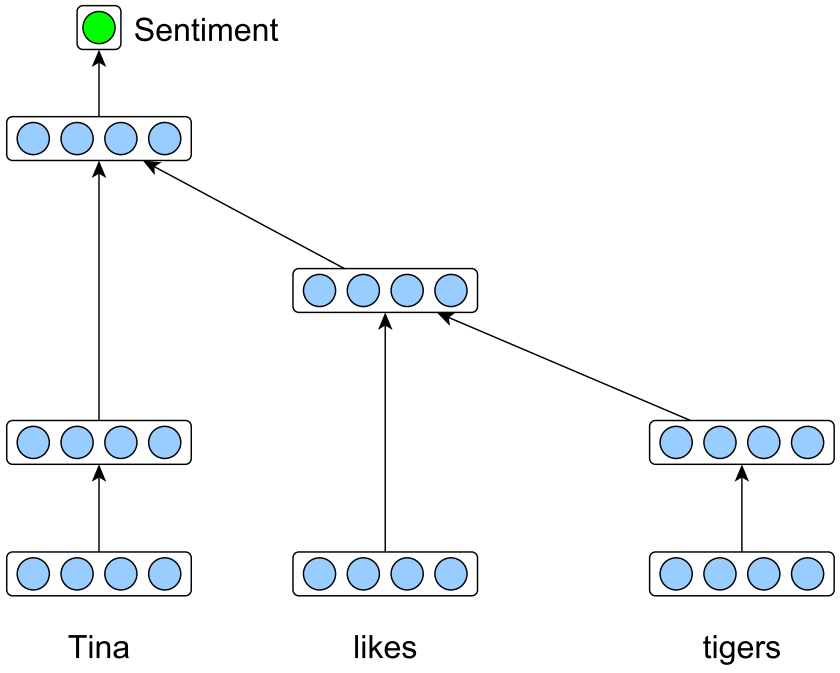
Convolutional Neural Networks were designed for image classification but can be adapted to work for language, too.

Recursive Neural Networks are a language-centric variant that have found success particularly for classification tasks.

Recursive Neural Networks

Composition follows syntactic structure

- Accumulation of state follows syntax
- Can chose any form of tree structure



Composition Function

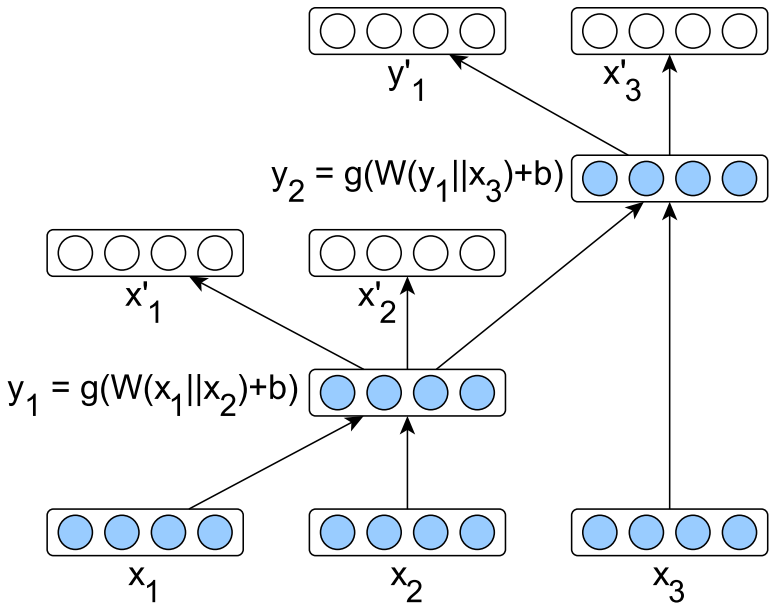
$$y = g(W_l x_l + W_r x_r)$$

A simple composition function as above is often sufficient. However, it is also possible to substitute this for a recurrent cell.

Recursive Neural Networks

While recursive networks have no simple generative counterpart, it is possible to improve classifier training by adding an additional autoencoder loss.

Autoencoder Signals



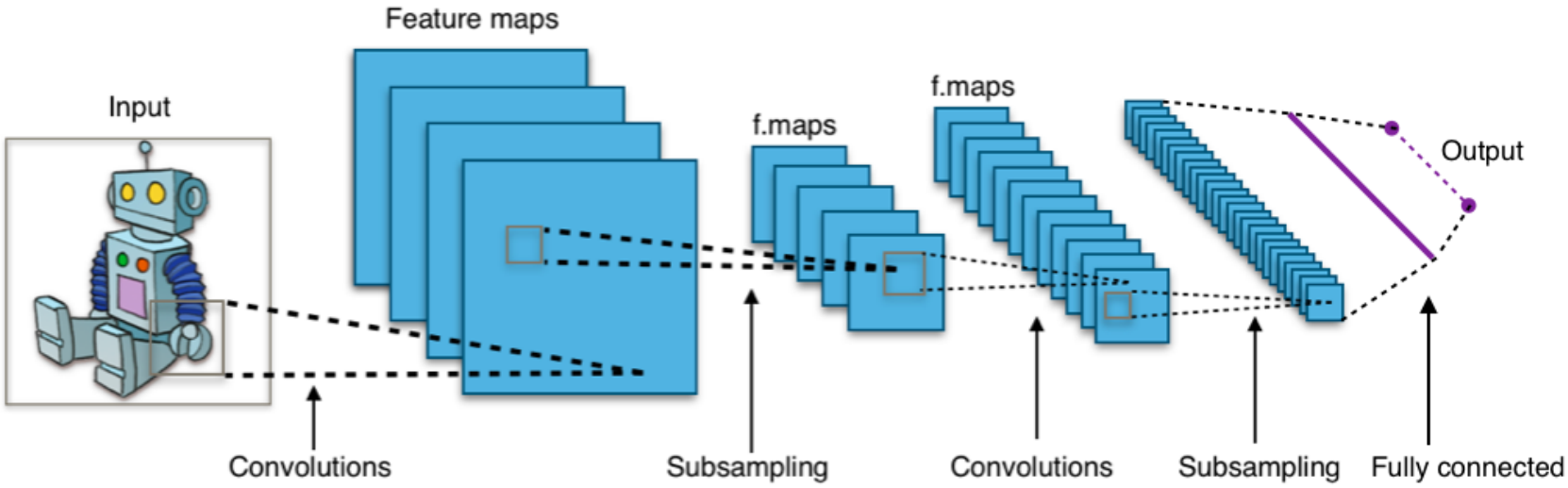
Reconstruction Error

$$E_{rec}(i,\theta) = \frac{1}{2}\left\| x_i - x_i' \right\|^2$$

This learns a compression function over the input space.

Convolutional Network Networks

For images Convolutional Neural Networks are very useful:

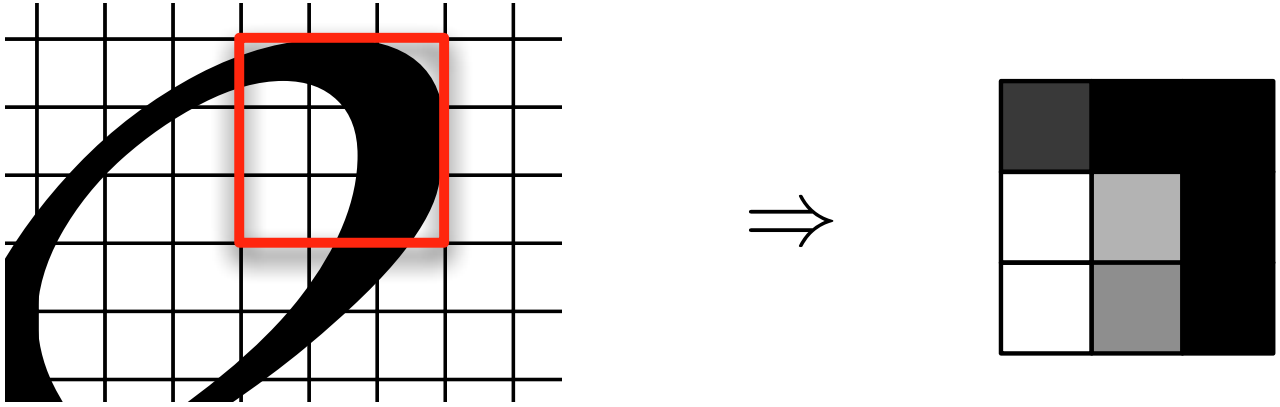


Convolution iteratives takes a matrix of inputs and computes a (smaller) matrix of outputs

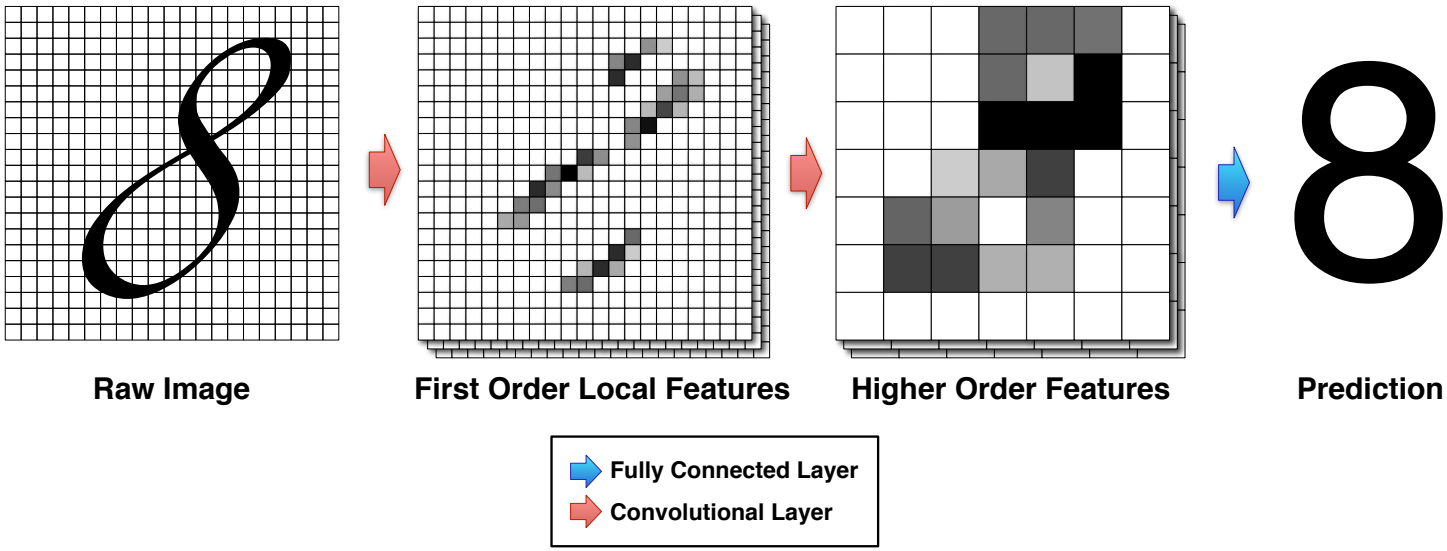
Subsampling takes a matrix of inputs and **pools** those into a single element , e.g. by taking the maximum

Convolutional Neural Networks

Convolutional window acts as a classifer for local features.



Stacked convolutional layers learn higher-level features.



Convolutional Neural Networks

Reasons to consider CNNs for Text

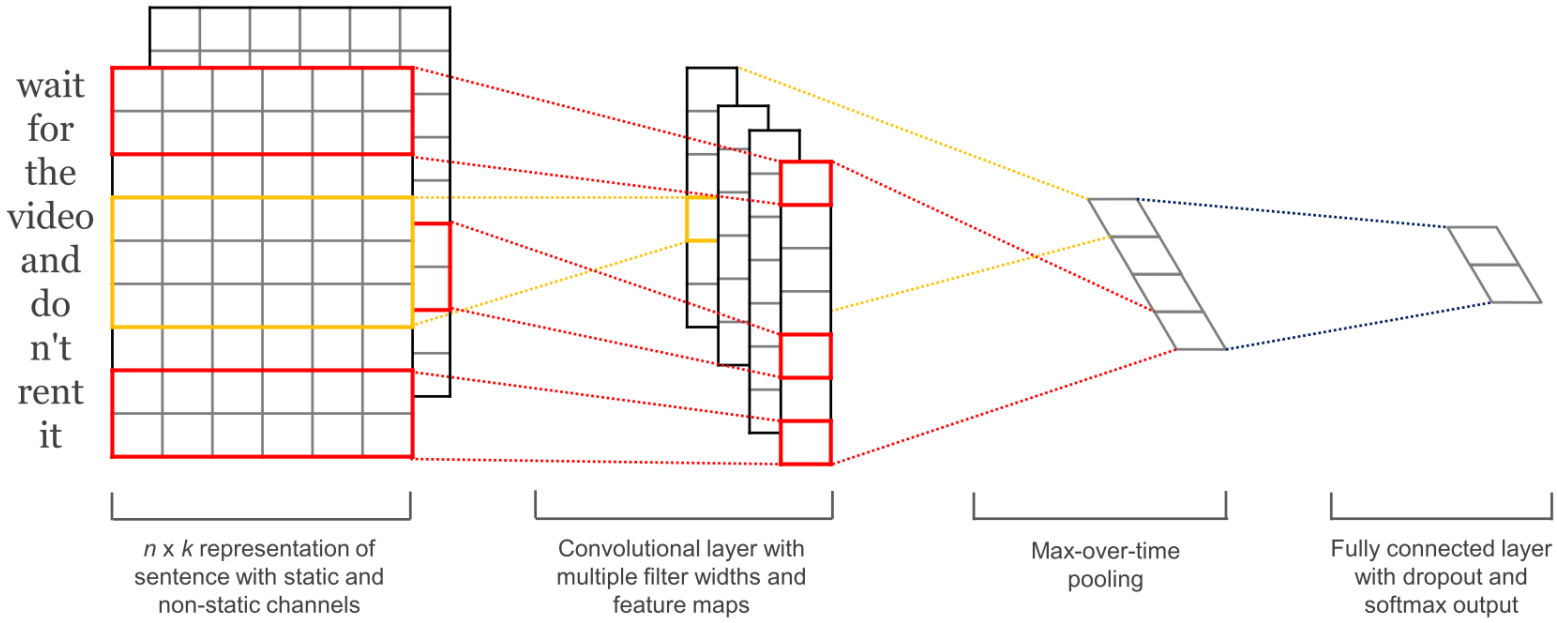
✔ Really fast (GPUs! See Thursday)

✔ BOW is often sufficient (see Naïve Bayes)

✔ Actually can take some structure into account

✘ Not sequential in its processing of input data

✘ Easier to discriminate than to generate variably sized data



Convolutional Neural Networks

Assume your data is of shape $\mathbb{R}^{M \times N \times K}$, where M is the number of input words, N the size of the input embeddings and K the number of feature maps (initially 1). Let x_i denote a layer in the CNN.

Convolutional Layer

Each filter $f_{l,m}$ is an dot product applied across the input data ($x_l \in \mathbb{R}^{M_l \times N_l \times K_l}$). s denotes a given segment of the input map.

$$o_{f_{l,m},s} = W_{f_{l,m}} \odot x_{l,s}$$

Key decisions are the number of filters (K), size of the filter (W_f) and stride with which the filter moves across the input.

Max-Pooling Layer

$$x'_{ijk} = \max\{x_{i'j'k} : i \leq i' < i + p, j \leq j' < j + p\}$$

Actual maths will look different for efficiency purposes. Gradients will get covered in Thursday's lecture or calculate yourself - reasonably simple.

End

Thursday's lecture will discuss GPUs for Neural Networks and give more details about convolutional networks.
Going forward we will build on the basic classifiers discussed today for more complex architectures and problems.

Sources and Further Reading

http://cs231n.github.io/convolutional-networks/
http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Lai et al. (2015) "Recurrent Convolutional Neural Networks for Text Classification"
Hermann (2014) "Distributional Representations for Compositional Semantics"
Kalchbrenner et al. (2014) "A Convolutional Neural Network for Modelling Sentences"
Socher et al. (2012) "Semantic compositionality through recursive matrix-vector spaces"