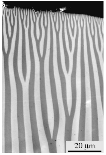
Learning from the Classics

Handwriting Generation using RNNs

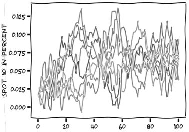
Thomas Viehmann
tv@mathinf.eu

Hacking Machine Learning Munich, 26 June 2018

About me



- Thomas Viehmann
- Mathematical modeller
- Ph.D. in Mathematics (Bonn) – Mathematical proof of fractal behaviour in a model for magnets



- Actuary and consultant for 9 years - helping insurance companies with their maths for financial and risk modelling, statistics etc.
- Hacker: debian Developer emeritus, contributed some 30 features and bugfixes to PyTorch
- Founded consultancy MathInf GmbH in May 2018.
- Core ML interests: Models that are aware of uncertainty, explaining model outputs, NLP, GANs, how to learn and teach AI
- ML blog: `https://lernapparat.de/`

About MathInf

Mission:

Helping companies build better AI through mathematical modelling

Make AI reliable:

- Models that are aware of uncertainty
- Explaining model outputs

  $\rightarrow$ more details another day

Modelling focus:

- Natural Language Processing
- Customizing models from various domains
- General statistical modelling (e.g. for insurance)
- "Classical" actuarial / financial modelling

`https://mathinf.eu/`

Why Handwriting Generation?

Handwriting fonts only go so far

Why Handwriting Generation?

Handwriting fonts only go so far

I like computers doing stuff

Why Handwriting Generation?

MathInf

Handwriting fonts only go so far

I like computers doing stuff

So what does it take to make the computer write?

Handwriting Generation

Article: Alex Graves, *Generating Sequences With Recurrent Neural Networks*, `https://arxiv.org/abs/1308.0850`

This is 5 years old, why study this?

Handwriting Generation

Article: Alex Graves, *Generating Sequences With Recurrent Neural Networks*, `https://arxiv.org/abs/1308.0850`

This is 5 years old, why study this?

- Instructive example for probabilistic modelling for training / prediction
- Much simpler than Seq2Seq etc. but has many of the important techniques
- Very simple attention model

→ **Great insights / chores ratio**

Graves's paper also discusses text generation as made very popular by A. Karpathy's *Unreasonable Effectiveness of RNNs* blog post.

Dataset

Typical dataset: IAM Online Handwriting Database.[1], 9950 lines

Online = We get a series of coordinates of the strokes as they are written, rather than a picture of the handwriting itself.

Preprocessing:

- Instead of strokes and absolute coordinates, convert them to (relative) pen movements and a flag (pen up / pen down).
  $\rightarrow$ makes the series stationary
- Some mild cleaning
- Standardize to mean 0 and standard deviation 1 in x and y (separately).

Text: I like computers....

Stroke: ($\sim$ 700 rows)

x	y	pen
-0.20	-0.00	0
0.16	0.68	0
-0.20	0.19	0
-0.20	0.41	0
-0.23	0.66	0
-0.27	0.73	0
-0.28	0.91	0
-0.30	0.98	0
-0.30	1.04	0
-0.30	1.02	0
-0.30	0.97	0
-0.31	0.88	0
-0.29	0.75	1
6.26	-7.52	0
-0.24	0.21	0
...	...	...

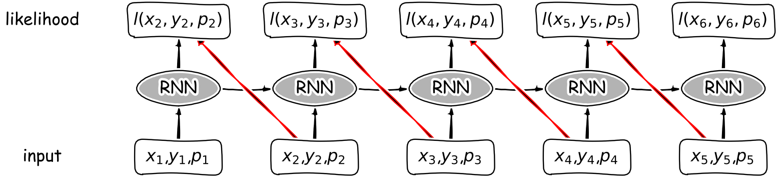
[1] http://www.fki.inf.unibe.ch/databases/iam-on-line-handwriting-database

Input and Output for Training and Prediction

For sequence-generating RNNs, the distinction between training and prediction becomes more apparent:

Training

score next output based on model density (loss = negative log likelihood)

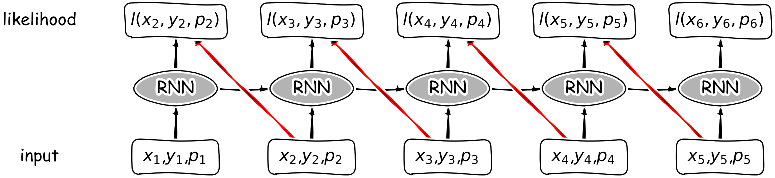


Input and Output for Training and Prediction

For sequence-generating RNNs, the distinction between training and prediction becomes more apparent:

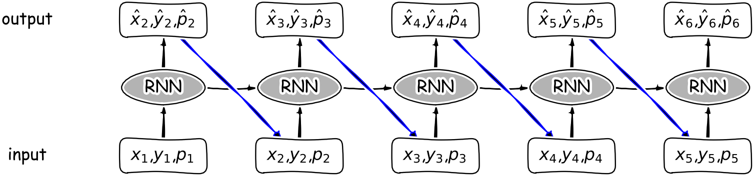
Training

score next output based on model density (loss = negative log likelihood)



Prediction

draw sample from model distribution and feed as next input



Loss functions?

Real valued data x, y – could we just use squared Euclidean distance?

How would the predictions look like?

What to do with the pen?

Loss functions?

Real valued data x, y – could we just use squared Euclidean distance?

How would the predictions look like?

What to do with the pen?

Enter probabilistic modelling:

Instead of directly outputting quantities, use NN to output parameters of probability distributions.

Joint normal distribution for x, y.

Pen as a Bernoulli variable with probability p

$\Rightarrow$ training: negative log likelihood; prediction: sample

Loss functions?

Real valued data x, y – could we just use squared Euclidean distance?

How would the predictions look like?

What to do with the pen?

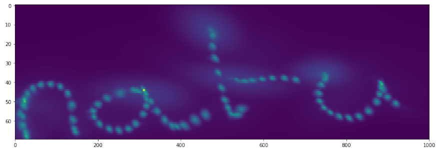
Enter probabilistic modelling:
Instead of directly outputting quantities, use NN to output parameters of probability distributions.

Joint normal distribution for x, y.

Pen as a Bernoulli variable with probability p

$\Rightarrow$ training: negative log likelihood; prediction: sample

Final twist: Use blend of Gaussian distributions with weights also given by NN, *Mixture Density Networks*, to capture different modes (within letter, next letter, next word).

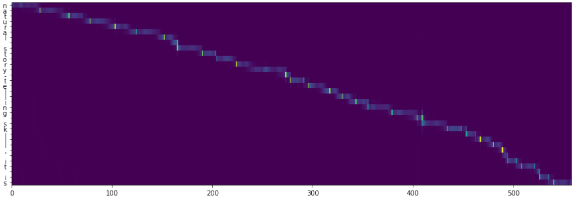


Superimposed density plot

Attention

So far, we haven't talked about what we want to write![2]
Typical thing to do: Take one-hot encoded sequence of characters.
Cannot feed it all at once and the timestep is *not* the character.
$\Rightarrow$ Use **attention mechanism**[3] - RNN looks at one character at a time:

- Position i starting with $i = 0$. Feed character at i to the RNN
- RNN in turn outputs how much to advance i for next prediction
- (use soft version to enable gradient descent and a mixture model)



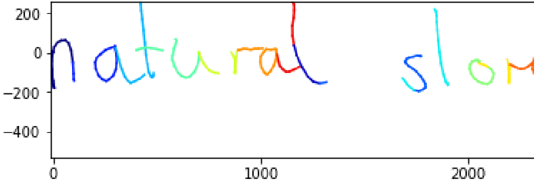
x-axis: time (points)
y-axis: i word attention

[2]Indeed, Graves also does "freestyle" (unconditioned) handwriting in the paper.
[3]This is a bit different from "query-based" attention that is a cornerstone of modern sequence processing.

Attention

So far, we haven't talked about what we want to write![2]
Typical thing to do: Take one-hot encoded sequence of characters.
Cannot feed it all at once and the timestep is *not* the character.
$\Rightarrow$ Use **attention mechanism**[3] - RNN looks at one character at a time:

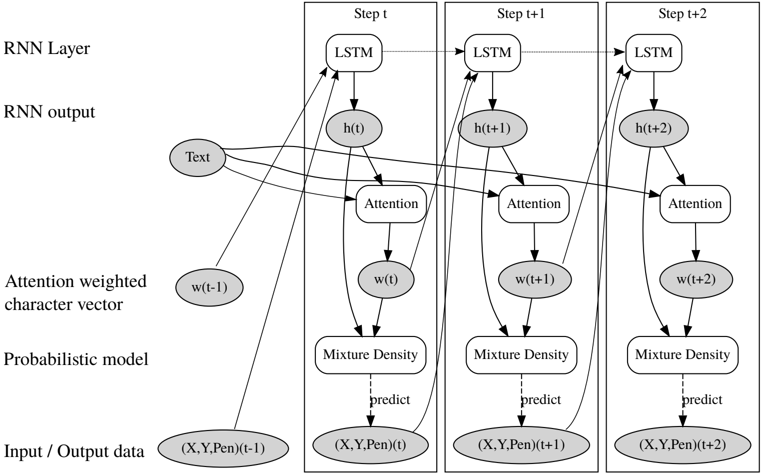
- Position i starting with $i = 0$. Feed character at i to the RNN
- RNN in turn outputs how much to advance i for next prediction
- (use soft version to enable gradient descent and a mixture model)



sample output with (peak) i coded as the color

[2] Indeed, Graves also does "freestyle" (unconditioned) handwriting in the paper.
[3] This is a bit different from "query-based" attention that is a cornerstone of modern sequence processing.

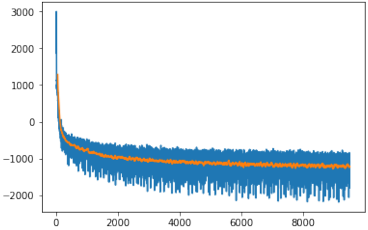
Putting it all together



Graves also has a three layer model with Bayesian regularization (Variational Bayesian in today's terms).

Training

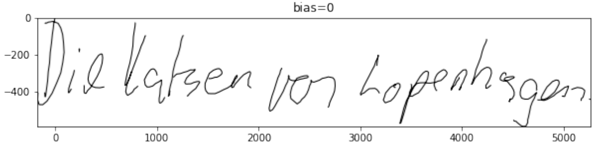
Standard technique: "Teacher forcing" - feed target sequence as inputs rather than actual output.



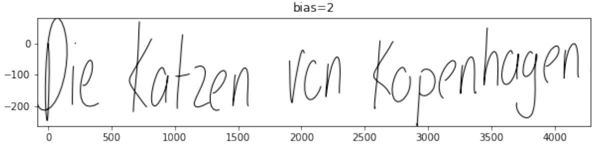
- Smoothed loss is ~ 1200, similar to what Graves reports.
- I ran the model for 50 epochs. Each epoch takes 4.5 minutes on a GTX1080, so < 4 hours total.
- Source (Jupyter Notebook with PyTorch) and pretrained model available at `https://lernapparat.de/handwriting-generation-rnns/`

Enjoy

Let the model draw the text you give it:



You can bias the predictions towards their mean value to get “cleaner” handwriting:



Graves has more examples, including “priming”: Feed a bit of training input first, then the RNN will imitate the style of the training input in further predictions.

Summary

In implementing the handwriting generation RNN, we used

- typical RNN setup for training / prediction,
- probabilistic modelling,
- a prototype of attention.

Great things to try out:

- weight dropout / Variational Bayes techniques to mimic MDL-regularization,
- multi-layer RNN,
- extend to SketchRNN – which is similar to a typical seq2seq model with encoder and decoder but uses many similar ideas as the handwriting RNN.

Do checkout the original Graves paper, it is very well written.

MathInf

Thank you!

Your questions and comments

Source code and slides at
https://lernapparat.de/handwriting-generation-rnns/