Assignment 4: Recurrent Neural Network for text synthesizing

—

Project Report

–

DD2424

Fabian Assarsson

2018-06-10

Executive Summary

We implement and evaluate a recurrent neural network ('RNN') with a hidden state consisting of m tanh-non-linearities, a softmax output activation function and a smoothed cross-entropy loss. We employ the AdaGrad algorithm to efficiently train the network through a gradient-based learning. By training, validating, and testing on a Harry potter Dataset, we subjectively evaluate different synthesized text output.

Contents

1 Introduction

1.1 Notation

We employ much the same notation as is seen in Deep Learning[**Goodfellow-et-al-2016**].

Numbers and Arrays

a – denotes a scalar, integer or real
$\boldsymbol{a}$ – denotes a vector
$\boldsymbol{A}$ – denotes a matrix
$\mathbf{A}$ – denotes a tensor
$\boldsymbol{I}$ – the identity matrix
diag($\boldsymbol{a}$) – denotes a square, diagonal matrix with entries given by $\boldsymbol{a}$
a – a scalar random variable
$\mathbf{a}$ – a vector-valued random variable
$\mathbf{A}$ – a matrix-valued random variable

Indexing

a_i – Element i of vector $\boldsymbol{a}$, indexed from 1.
a_{-i} – All elements from $\boldsymbol{a}$, except i
$A_{i.j}$ – Element i,j of matrix $\boldsymbol{A}$
$\boldsymbol{A_{i,:}}$ – Row of matrix $\boldsymbol{A}$
$\boldsymbol{A_{:,i}}$ – Column of matrix $\boldsymbol{A}$
$A_{i,j,k}$ – Element i,j,k of tensor $\mathbf{A}$
$\mathbf{A}_{:,:,\mathbf{i}}$ – 2D slice of a 3D tensor
a_i – Element i of random vector $\mathbf{a}$

Linear Algebra Operations

$\boldsymbol{A}^\top$ – Transpose of matrix $\boldsymbol{A}$.
$\boldsymbol{A}^+$ – Moore-Penrose Pseudoinverse of $\boldsymbol{A}$
$\boldsymbol{A} \odot \boldsymbol{B}$ – Hadamard product between $\boldsymbol{A}$ and $\boldsymbol{B}$
det($\boldsymbol{A}$) – The determinant of $\boldsymbol{A}$

Calculus

$\frac{dy}{dx}$ – derivative of y w.r.t x
$\frac{\delta y}{\delta x}$ – partial derivative of y w.r.t x.
$\nabla_{\boldsymbol{x}} y$ – gradient of y w.r.t $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$ – matrix derivates of y w.r.t $\boldsymbol{X}$
$\nabla_{\mathbf{X}} y$ – tensor containing derivates of y w.r.t $\mathbf{X}$

Probability and Information Theory

a⊥b – The random variables a and b are independent
a⊥b|c – They are conditionally independent given c
P(a) – A probability distribution over a discrete variable a
p(a) – A probability distribution over a continuous variable a
a $\sim P$ – random variable a has distribution P
$\mathbb{E}_{x\sim P}[f(x)]$ or $\mathbb{E}f(x)$ – Expectation of $f(x)$ with respect to $P(x)$
Var($f(x)$) – Variance of $f(x)$ under $P(x)$
Cov($f(x)$,$g(x)$) – Covariance of $f(x)$ and $g(x)$ under $P(x)$
H(x) – Shannon Entropy of the random variable x
$D_{\mathrm{KL}}(P||Q)$ – Kullback-Leibler divergence of P and Q
$\mathcal{N}(\mathbf{x};\mu,\mathbf{\Sigma})$ – Gaussian distribution over $\mathbf{x}$ with mean μ and covariance $\mathbf{\Sigma}$

1.2 The RNN

In our fourth assignment, we have implemented a recurrent neural network. It takes two input sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$, where $\boldsymbol{Y}$ is $\boldsymbol{X}$ shifted one step in time. For each character in our $\boldsymbol{X}$, we want to predict the corresponding character in $\boldsymbol{Y}$. As $\boldsymbol{Y}$ is shifted one step, we are essentially trying to learn the next character in the sequence we are providing. This setup is generally known as *sequence to sequence learning* and is one of the main uses of an RNN. After training, we provide the model with one character and randomly select the next one according to the learned distribution for each character defined by our softmax-output-layer. In this scheme, we have a stochastic output that mimics the probability distribution over our character space, and we hope that it is non-uniform enough as to produce believable text outputs. It should be noted that the RNN has no formal prior assumptions about language inherent in the model, so it has no explicit understanding of grammar, word-lengths and legal/illegal character sequences in language. The assumption is that a large enough correct corpus, coupled with enough training time will make the distributional properties of our output layer to implicitly mimic this understanding. This could be positioned in contrast to models that explicitly makes use of language understanding through lemmatization, stemming or word2vec-representations.

1.2.1 The tanh activation

When agreeing on the fact that deep network have several benefits over shallow and wide counterparts (i.e. shallow networks that have the same number of computed parameters), we need to choose a non-linearity in this transformation as well. While the softmax makes sense for the output layer in terms of the probabilistic and information theoretic interpretations discussed below, the intermediate non-linearity must be chosen according to some other criterias. Now, in earlier reports, we have argued that the tanh-function is worse than the ReLu-function, so it might seem strange that we have introduced it in our RNN-setup. If one views a RNN "unrolled" it can be thought of as a very deep

feed-forward network with shared weights between the layers. The nature of the ReLu-function makes it explode in it's positive direction, whereas a tanh-function is smoother across it's range and has a non-monotonic derivative. It therefore helps with the exploding gradients problem.

1.2.2 The Softmax activation

Given that our dataset is "fixed" and that we do not preprocess it in any way, and given that our parameters are initialized as

$$\begin{aligned} \boldsymbol{W} &\sim \mathcal{N}(\mathbf{x};\mu,\boldsymbol{\Sigma}), \mu = \mathbf{0}, \boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}, \sigma = 0.1 \\ \boldsymbol{b} &\sim \mathcal{N}(\mathbf{x};\mu,\boldsymbol{\Sigma}), \mu = \mathbf{0}, \boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}, \sigma = 0.1 \end{aligned} \tag{1}$$

we can't be sure of the norm of the output from our affine transformation. In order to be a proper probability distribution, the values need to be normalized and as some values can be negative it makes intuitive sense to tream them as they where log probabilities. To recover the actual probabilities, we therefore need to take the exponent of all outputed values and then normalize. It turns out that this reasoning leads us to the very definition of the softmax non-linearity function:

$$\text{softmax}(\boldsymbol{s}) = \frac{\exp(\boldsymbol{s})}{\mathbf{1}^\top \exp(\boldsymbol{s})} \tag{2}$$

1.2.3 Cross-Entropy Loss

As we have decided to interpret the output of our affine transformation as unnormalized log probabilities of class membership and then used the softmax activation to normalize and recover, we now have a "proper" probability distribution $\hat{P}$ of class membership outputted by our network. If we use a one-hot encoding for each $y_i \in \{1, ..., K\} \rightarrow \boldsymbol{y}_i = \boldsymbol{e}_j$ when y_i takes on value j, we can view our truth-labels as belonging to a different distribution P. It makes sense to try to make these distributions as close to each other as possible, but as regular euclidean distances doesn't apply in probabilistic settings we need to introduce the *cross-entropy loss*. As cross-entropy between $\hat{P}$ and P can be expressed as:

$$H(P, \hat{P}) = H(P) + D_{\text{KL}}(P||\hat{P}) = \mathbb{E}_P[-log\hat{P}] \tag{3}$$

We that, not only do we do an analogous "distance minimization" (by minimizing the Kullback-Leibler divergence), but we also have a log-term that gets canceled out by our softmax for much easier and nice-looking gradients! The cross-entropy loss therefore makes sense in two ways; it is easy to interpret and calculate and it's intuitive to want to minimize the encoding cost between our "true distribution" given by our labels, and our guessed distribution given by the network. Minimizing the cross-entropy loss is also equivalent to performing a maximum likelihood estimation of our data. A reasonable assumption about

our data is that it is independent and identically distributed, giving rise to a joint likelihood function $\mathcal{L}$ of the form:

$$\mathcal{L} = \prod_n \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \tag{4}$$

The likelihood is the probability which our model gives to our correct class. With one-hot encoding this reduces to the dot product between our ground truth one-hot label and our guessed distribution. If we want to maximize this function, it is the same as minimizing it's negative log (since the logarithm is monotonic):

$$-\log\mathcal{L} = -\sum_n \log\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \tag{5}$$

or, with one-hot encoding after some manipulation is equal to our cross-entropy loss:

$$-\log\mathcal{L} = -\sum_n \log(\boldsymbol{y}^\top \boldsymbol{p}) \tag{6}$$

1.2.4 Regularization

Regularization of this network, with an L_2-penalty on the weight parameters, will punish models that have large weight parameters and reward those with small, diffuse ones.

We have already stated that our cross entropy function is performing a maximum likelihood estimation ('MLE'). But, if we want to go "fully Bayesian", we would like to perform a maximum a posteriori estimation of our parameters given our fixed data. This results in us needing to choose a suitable prior distribution over our parameters, namely the conjugate prior to our likelihood. As the conjugate prior to a gaussian is another gaussian, we can change the likelihood maximization objective to a maximum a posteriori instead:

$$\text{MAP} = \prod_n \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})\, \mathcal{N}(\boldsymbol{W}; \boldsymbol{0}, \boldsymbol{\lambda}^{-1}) \tag{7}$$

Which, under the same argument as above, reduces to:

$$-\log \text{ MAP} \propto -\sum_n \log(\boldsymbol{y}^\top \boldsymbol{p}) - \boldsymbol{\lambda}\boldsymbol{W}^2 \tag{8}$$

Consequently, introducing a L_2-regularization on the weight matrix $\boldsymbol{W}$ is the same as performing a MAP-estimate of our parameters when we assume a Gaussian prior on the distribution of our weights. It should be noted here that it doesn't imply that we 'create' our weight-matrix as a gaussian (which we do), but rather that our "guess" on the final form of the learned matrix, before any evidence, is that it is Gaussian.

1.3 Gradient calculations and verifications

The gradient calculations was performed correctly. To ensure this, they where checked against ComputeGradsNumSlow as provided with the relative error method from the assignment. I picked the largest differing value and ensured that it was around 10^{-7}, an arbitrary threshold value. The actual result, for 25 examples was:

$$\begin{aligned} \epsilon_b &= 1.7467 * 10^{-9} \\ \epsilon_c &= 3.0503 * 10^{-10} \\ \epsilon_U &= 2.8536 * 10^{-9} \\ \epsilon_W &= 1.9630 * 10^{-7} \\ \epsilon_V &= 8.2572 * 10^{-9} \end{aligned} \tag{9}$$

2 Results

As we train the model, we can see an overall reduction of the smoothed loss. It should be noted that the training quite quickly saturates and then oscillates between 39-42 in error value. Observing the error development over time, it is reasonable to believe that two parts of the dataset is especially hard to learn, giving rise to some confusion for the model and effectively hindering further error reduction/learning. Below are the graphs of the error across 6 epochs.

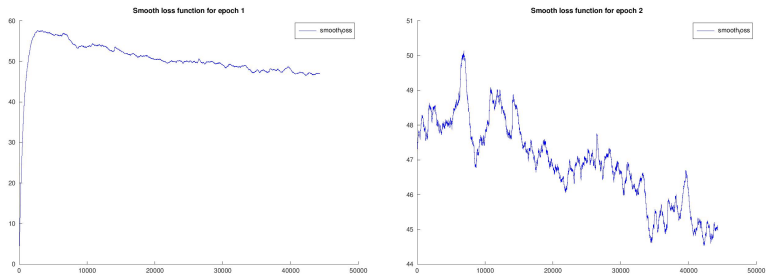

Figure 1: Left: Epoch 1. Right: Epoch 2

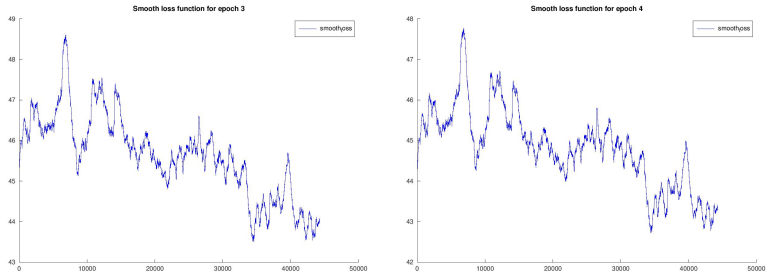

Figure 2: Left: Epoch 3. Right: Epoch 4

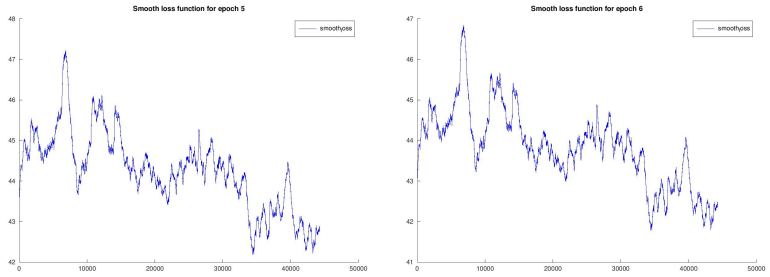

Figure 3: Left: Epoch 5. Right: Epoch 6

2.1 Evolution

In increments of 10.000, this is how my model evolved:

> theedtjbidcthe sin ca" tlstbe sid bt"d tSos ss .iflrets.Fsd "h nsoatry aolpoulre cong.tea vicntuan uot voiler aain "wt nrhs odd-h oysdude ge tserpthedoss cwedy lont atatchucy.rispoMisy deee s ael t
>
> to. Weaso. Cothe gexincered, oktore Copcled the, shatrecocht anw rusmey rusary erf thein, Dack thith spell, datcwartesizisiy thuthtitht of it vears't sark santerly domting masper,'s Clumt Ducking bat
>
> it do a reablige able the quacing air Sisly. Is staye, folmil of thith yoly eveupfee kean-o veay, Tarened eperoth aid meaivey he were furiigbeots of ceridt," ous berod, the nouds, agmiind Harry't in
>
> ough any maid widr il eve ster it. "Are go be hist on theiny to onled of laks woutly nowverd coofeed sing he wal . brea are nirsing. . Golled then ople, Ron amp he the lee stuagn, gesteasolouren nome
>
> l wit even a pulked had the stant it who gisur to Harry.. He wald fot prieved fool elders. ." Hardy - tipped bicy't withe've veamlf read in goodt the lat to ne, his ulllild thinut was to not to ckeas
>
> con, know a reing alltroNg up alow, and of fage. . . Avirgarble theored Gown ost and in give anotsermarir Rom tirt to a rotl foupred anoiment shinquonasen" in dizeated tho romtilion to lowh hid panty
>
> ed supbe of how him warr, of thearen're ewceAnter? Harring him motlitlikny hag jespubfeoulving has it any if cway whiwe thought his at Harmigbled of Harry dows, but tast, whon's in theing we pore nig
>
> lys at my Durred strost frould the ligge the hapce to sut had mering out beronts over but tot he brecing. "Harry funjuppering the cweritint id end mupiese town, and grouces ne soy and pownd sec and ae
>
> bare up at thazh resnduth a toldemeswes ever of Vark voughed to Mat Voldesring ae Hegming the sind is tho atkyented at growts intain, to caving his at Eagh ingast; Weapserted Exeakne anonvold mark.
>
> iched Ibes, and lastly. Harrys you his look," said Hermionedle enore tean erinting they was exyoug; Hersing his hear bepiann not of kiyss -DoYTH. Deading his mut oneed. "Whapingied wind as recken't

2.2 Harry Potter: Redux

And this glorious 1000-character output is the final result of my best model:

> The besincort is Crouch, and onfory, was had gate and at the about again, broor him Dork as olend the moushted, yeirs let arounds a way foortled was core! "minved to walled a straiked, wann." You smok other; matenthta ruds I had digging is chann Howlit mave to feets. He conamemosily gray ampicole. It acrevere of millionessay, youre though me to Veld loyst he hour id but Harry's his any you tore the pupped quil. "The tor, returd wrooned Wormsalfor?" A thered tourds. Jowly up tiff the boully. "That halss what the foor's, him. The sood ast tow. 'y paider," said Cedene, Owhc flater!" It wat in calftaik, his as the recarts - sone that yaur, and the'r' leasly told and chose stiltitigst at the had haved with his list be pyot up forting made actaph othave. "And the pucdine" said Spich fying the now serly; Shh's been to, but where howing for arsul beens please and I wazing Malden" groce twarat of when Maxbens Dumbledore get for head aroung a looked awall, you pretend; their lite heats