Michael Jahrer
Topic Author
1st place

1st place with representation learning

Posted in porto-seguro-safe-driver-prediction 5 years ago

1015

Thanks to Porto Seguro to provide us with such a nice, leakage-free, time-free and statistical correct dataset.
A nice playground to test the performance of everything, this competition was stat similar to otto, like larger testset than train, anonymous data, but differ in details.
I wanna dive straight into solution. Its a blend of 6 models. 1x lightgbm, 5x nn. All on same features, I just removed *calc and added 1-hot on *cat.
All neural nets are trained on denoising autoencoder hidden activations, they did a great job in learning a better representation of the numeric data. lightgbm on raw data.
Nonlinear stacking failed, simple averaging works best (all weights=1). Thats the final .2965 solution.
2 single models would have been enough to win (#1 + #2 give me 0.29502 on private).
The complete list of models in the final blend:

nr	feat	Norm-alization	unsupervised		model		5-fold CV		kaggle	
			type	time[h]	type	time[h]	gini	logloss	public	private
1	f0	-	-	0	**lightgbm** objective=binary, 1400 rounds, boosting_type=gbdt, learning_rate=0.01, max_bin=255, num_leaves=31, min_data_in_leaf=1500, feature_fraction=0.7, bagging_freq=1, bagging_fraction=0.7, lambda_l1=1, lambda_l2=1	0.13	0.28843	0.15163	0.28368	0.29097
2	f0	Rank Gauss	**denoising autoencoder, deep stack** topology=221-1500r-1500r-1500r-221l, lRate=3e-3, minibatchSize=128, backend=GPU32, lRateDecay=0.995, inputSwapNoise=0.15, nEpochs=1000	5	**neural net** topology=4500-1000r-1000r-s, lRate=1e-4, lRateDecay=0.995, regL2=0.05, dropout=0.5, dropoutInput=0.1, minibatchSize=128, backend=GPU32, loglossUpdate=1, nEpochs=150	3.45	0.29036	0.15184	0.28970	0.29298
3	f0	Rank Gauss	**denoising autoencoder, deep stack** topology=221-1500r-1500r-1500r-221l, lRate=3e-3, minibatchSize=128, backend=GPU32, lRateDecay=0.995, inputSwapNoise=0.07, nEpochs=1000	5	**neural net** topology=4500-1000r-1000r-s, lRate=1e-4, lRateDecay=0.995, regL2=0.05, dropout=0.5, dropoutInput=0.1, minibatchSize=128, backend=GPU32, loglossUpdate=1, nEpochs=200	4.6	0.28942	0.15185	0.28846	0.29377
4	f0	Rank Gauss	**denoising autoencoder, deep stack** topology=221-1500r-1500r-1500r-221l, lRate=2.9e-3, minibatchSize=128, backend=GPU32, lRateDecay=0.995, inputSwapNoise=0.15, nEpochs=1000 colGroups=1	5	**neural net** topology=4500-1000r-1000r-s, lRate=1e-4, lRateDecay=0.995, regL2=0.05, dropout=0.5, dropoutInput=0.1, minibatchSize=128, backend=GPU32, loglossUpdate=1, nEpochs=76	1.8	0.29062	0.15174	0.28778	0.29265
5	f0	Rank Gauss	**denoising autoencoder, bottleneck** topology=221-15000r-15000r-3000l-15000r-15000r-221l, lRate=1e-3, minibatchSize=128, backend=GPU32, lRateDecay=0.995, inputSwapNoise=0.1, nEpochs=300	75.2	**neural net** Topology=3000-1000r-1000r-s, lRate=1e-4, lRateDecay=0.995, regL2=0.05, dropout=0.5, dropoutInput=0.1, minibatchSize=128, backend=GPU32, loglossUpdate=1, nEpochs=200	3.2	0.28904	0.15202	0.28745	0.29373
6	f0	Rank Gauss	**denoising autoencoder, deep stack** topology=221-1500r-1500r-1500r-221l, lRate=2.8e-3, minibatchSize=128, backend=GPU32, lRateDecay=0.995, inputSwapNoise=0.2, nEpochs=1000	5	**neural net** topology=4500-1000r-1000r-s, lRate=1e-4, lRateDecay=0.995, regL2=0.05, dropout=0.5, dropoutInput=0.1, minibatchSize=128, backend=GPU32, loglossUpdate=1, nEpochs=150	3.4	0.29091	0.15180	0.28856	0.29303
					linear blend, all w=1 of 1,2,3,4,5,6	0.01	0.29442	0.15159	0.29136	0.29653

Font is a bit small, you need to increase the zoom with ctrl (+).
The difference to my private .2969 score is I added bagging versions nBag=32 of the above mentioned 6 models, all weight=1, and Igor's 287 script with weight=0.05.
Was not really worth the efford for .2965 → .2969 gain huh!? I selected these 2 blends at the end.

feature engineering

I dislike this part most, my creativity is too low for an average competition lifetime, also luck plays huge role here. Therefore I like representation learning, its also an step towards AI.
Basically I removed *calc, added 1-hot to *cat features. Thats all I've done. No missing value replacement or something. This is featureset "f0" in the table.
This ends up in exactly 221 dense features. With single precision floats its 1.3GB RAM (1e-9*4221*(595212+892816)).
Thanks to the public kernels (wheel of fortune eg.) that suggest to remove *calc features, I'm too blind and probably would not have figured this out by myself.
I never remove features.

local validation

5-fold CV as usual. Fixed seed. No stratification. Each model has own rand seed in CV (weight init in nn, data_random_seed in lightgbm).
Test predictions are arithm. averages of all fold models. Just standard as I would use for any other task.
Somebody wrote about bagging and its improvements, I spend a week in re-training all my models in a 32-bag setup (sampling with replacement).
Score only improved a little.

normalization

Input normalization for gradient-based models such as neural nets is critical. For lightgbm/xgb it does not matter.
The best what I found during the past and works straight of the box is "RankGauss".
Its based on rank transformation. First step is to assign a linspace to the sorted features from 0,1, then apply the inverse of

its based on rank transformation. First step is to assign a linspace to the sorted features from 0..1, then apply the inverse of error function ErfInv to shape them like gaussians, then I substract the mean.
Binary features are not touched with this trafo (eg. 1-hot ones).
This works usually much better than standard mean/std scaler or min/max.

unsupervised learning

Denoising autoencoders (DAE) are nice to find a better representation of the numeric data for later neural net supervised learning.
One can use train+test features to build the DAE. The larger the testset, the better :)
An autoencoder tries to reconstruct the inputs features. So features = targets. Linear output layer. Minimize MSE.
A denoising autoencoder tries to reconstruct the noisy version of the features. It tries to find some representation of the data to better reconstruct the clean one.
With modern GPUs we can put much computing power to solve this task by touching peak floating point performance with huge layers. Sometimes I saw over 300W power consumption by checking nvidia-smi.
So why manually constructing 2,3,4-way interactions, use target encoding, search for count features, impute features, when a model can find something similar by itself?
The critical part here is to invent the noise.
In tabular datasets we cannot just flip, rotate, sheer like people are doing this in images.
Adding gaussian or uniform additive / multiplicative noise is not optimal since features have different scale or a discrete set of values that some noise just didnt make sense.
I found a noise schema called "swap noise". Here I sample from the feature itself with a certain probability "inputSwapNoise" in the table above. 0.15 means 15% of features replaced by values from another row.
Two different topologies are used by myself. Deep stack, where the new features are the values of the activations on all hidden layers.
Second, bottleneck, where one middle layer is used to grab the activations as new dataset. This DAE step usually blows the input dimensionality to 1k..10k range.

learning with train+test features unsupervised

You might think I am cheating when using test features too for learning. So I've done an experiment to check the effectiveness of unsupervised learning without test features.
For reference I took model #2, public:0.28970, private:0.29298. With exactly same params it ends up in a slighty weaker CV gini:0.2890. public:0.28508, private:0.29235.
Private score is similar, public score is worse. So not a complete breakdown as expected. Btw total scoring time of the testset with this "clean" model is 80[s].

other unsupervised models

Yes I tried GANs (generative adversarial networks) here. No success. Since NIPS2016 I was able to code GANs by myself. A brilliant idea. Generated MNIST digits looked fine, CIFAR images not that. For generator and discrimiator I used MLPs.
I think they have a fundamental problem in generating both numeric and categoric data. The discrimiator won nearly all the time on my setups. I tried various tricks like truncation the generator output. Clip to known values, many architectures, learn params, noise vec length, dropout, leakyRelu etc.
Basically I used activations from hidden layers of the discrimiator as new dataset.
At the end they were low 0.28x on CV, too low to contribute to the blend. Havent tried hard enough.

Another idea that come late in my mind was a min/max. game like in GAN to generate good noise samples. Its critical to generate good noise for a DAE.
I'm thinking of a generator with feature+noiseVec as input, it maximizes the distance to original sample while the autoencoder (input from generator) tried to reconstruct the sample... more maybe in another competition.

neural nets

Feedforward nets trained with backprop, accelerated by minibatch gradient updates. This is what all do here.
I use vanilla SGD (no momentum or adam), large number of epochs, learning rate decay after every epoch.
Hidden layers have 'r' = relu activation, output is sigmoid. Trained to minimize logloss.
In bottleneck autoencoder the middle layer activation is 'l' = linear.
When dropout!=0 it means all hidden layers have dropout. Input dropout often improve generalization when training on DAE features.
Here a slight L2 regularization also helps in CV.
Hidden layer size of 1000 works out of the box for most supervised tasks.
All trained on GPU with 4-byte floats.

lightgbm

Nice library, very fast, sometimes better than xgboost in terms of accuracy. One model in the ensemble. I tuned params on CV.

xgboost

I didnt found a setup where xgboost adds something to the blend. So no used here in Porto.

blending

Nonlinear things failed. Thats the biggest difference to otto competition where xgb, nn were great stackers.
Every competition has its own pitfalls. Whatever.
For me even tuning of linear blending weights failed. So I stick with all w=1.

software used

Everything I've done here end-to-end was written in C++/CUDA by myself. Of course I used lightgbm and xgboost C interface and a couple of acceleration libs like cuBLAS.
I'm a n00b in python or R like you guys are experts. My approach is still oldschool and low level. I want to understand what is going from top to bottom. At some time, I'll learn it, but currently there are just too much python/R packages that bust my head,

I'm stick with loop-based code.

hardware used

All models above can be run on a 32GB RAM machine with clever data swapping. Next to that I use a GTX 1080 Ti card for all neural net stuff.

total time spend

Some exaflops and kilowatts of GPU power was wasted for this competition for sure.
Models run longer than I spend on writing code. Reading all the forum posts also costs a remarable amount of time, but here my intention was dont miss anything. At the end it was all worth. Big hands to all the great writers here like Tilli, CPMP, .. really great job guys.

what did not work

upsampling, deeper autoencoders, wider autoencoders, KNNs, KNN on DAE features, nonlinear stacking, some feature engineering (yes, I tried this too), PCA, bagging, factor models (but others had success with it), xgboost (other did well with that) and much much more..

thats it.
The attatched file is the console log of model nr 3. If you do a "cat nn.cfg.log" from a Linux console you get the output with nice colors.

nn.cfg.log

Comments (441)

Sort by Hotness

Alan Jo • 6 months ago

2

Thank you for sharing

Yong Kang Chia • 2 years ago

1

Thanks for sharing! you being featured in this medium article https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d really improved my understanding of Gradient boosting and XGBoost!

Blesson Densil • 2 years ago

3

This is Great work. Thanks for sharing

Moore • 2 years ago

1

Congrats and Thanks for sharing. Very helpful.

Sude ADIGUZEL • 2 years ago

1

Thanks for sharing your knowledge!!

Umar Fathurrohman • 3 years ago

1

Thank you Michael, I just recently started learning Autoencoders.

I have two simple questions, since I have little experience on data science:

1. Could you show me how to read your topology? For example, how do I interpret topology=4000-1000r-1000r-s?
2. Suppose that I have 1000 rows and 100 features. Does this mean 15 features will have their values swapped? If yes, how many rows should I swap on each selected features?
   Quoting from above:

> Here I sample from the feature itself with a certain probability "inputSwapNoise" in the table above. 0.15 means 15% of features replaced by values from another row.

Thank you.

Michael Jahrer Topic Author • (1st in this Competition) • 3 years ago

4

no problem. to answer your questions:
1) 4000-1000r-1000r-s means 4000 input features, then two hidden layers of 1000 relu, then one output neuron sigmoid.
2) I always followed the approach of maximize the random samples drawn (max. entropy).
So on average 15 features out of 100 will be swapped by another values inside the feature column.

Umar Fathurrohman • 3 years ago

0

Ah I see now, thank you Michael!

Mehmet Öztürk • 3 years ago

thank you

1

Vinay Turpati • 3 years ago

Thanks!

3

Vinay Turpati • 3 years ago

Thanks @Michael Jahrer

3

piAI • 3 years ago

@mjahrer Thanks for sharing the detail approach and highlighting the role of representation learning.got to learn about several useful ideas.

1

Andy Harless • (21st in this Competition) • 5 years ago

Not just winning the competition by a huge margin, but also debunking two erstwhile truisms in one fell swoop:

- Deep learning only works with very large data sets
- Unsupervised learning isn't useful

54

o_to_hu • 3 years ago

Interesting

1

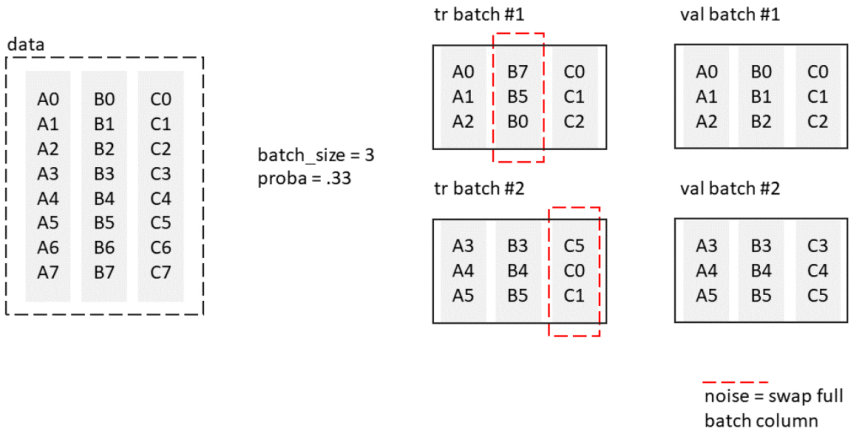
Selin Raja M • 3 years ago

Interesting

1

danzel • (1797th in this Competition) • 3 years ago

@mjahrer I've got two question regarding swap noise and data trafo.
Q1:
Do you sample and replace the "full" batch column - like it is shown in the picture below?



Q2:
Do you rankGauss transform all numerics (including raw cat features)?
So your feature set looks like:
rankGauss(numeric columns) + rankGauss(cat columns) + oneHot(binary columns)

Or is it:
rankGauss(numeric columns) + raw(cat columns) + oneHot(binary columns)

Cheers
danzel

Michael Jahrer Topic Author • (1st in this Competition) • 3 years ago

5

Hi Danzel,
[Q1] in my swapNoise idea I sample a single feature value with a specific probability from the same feature column. Its completely randomized within the batch matrix. Not just replace the full column with different values (as you showed).

You can try both swapping ideas (random per value [i,j], random for a full column [:,j]). But this just influences the autoencoder pretraining step. You need to verify if it works in a supervised setting later. I can say that I tried this dae+nn for many competitions, it only "worked" for two, here and in otto. I mean it improved in a very little numeric scale, if all.

[Q2]
The idea of rankGauss is to map the numeric feature values to a meaningful distribution based on ranking. Values which appears often in train get a bigger space after the rankGauss trafo (=ranking trafo). Based on my experience categoric columns are good as one-hot dense [0,0,0,1,0,0,0,0] representation for nn, this is what my normal pre-processing do. So in my rankGauss implementation I don't transform columns which has 2 unique values (assume these are from one-hot categories).
I think you mean raw cat features = integers ? This is bad, input values are too large. I recommend to preprocess any csv-based data:
rankGauss(numeric columns) + one-hot(cat columns) + (binary columns)
If cat columns has too many tokens, you can try using cnts or numeric integers instead.

danzel • (1797th in this Competition) • 3 years ago

2

Thanks for the quick reply!
__ edit

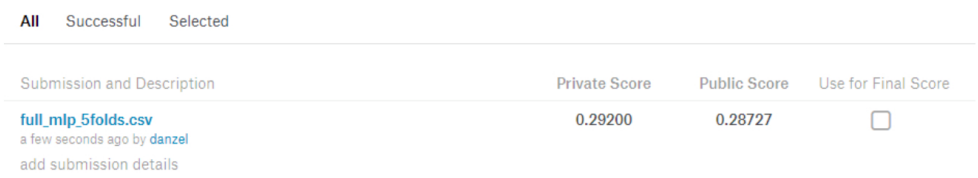
> in my swapNoise idea I sample a single feature value with a specific probability from the same feature column. Its completely randomized within the batch matrix. Not just replace the full column with different values (as you showed).

I replaced the full column with values sampled out of this exact column. I just wanted to know whether you add noise to the full input column (e.g. swap noise proba of .15 = 15 % of all columns got replaced by random data sampled from each of the columns distribution) or is it a more soft approach where 15 % of the columns contain a specific amount of noisy data.

danzel • (1797th in this Competition) • 3 years ago

12

So ... after trying for 6 months I finally made it (well ... almost).



Thanks @mjahrer for explaining all this without spoon-feeding everything :)
I've learned so much!

Language used:
R + keras

Hardware used:
NVIDIA GeForce GTX 1070 (8 GB)
+
32GB System-memory

https://media.makeameme.org/created/its-magic-drs8g8.jpg

Roberto Spadim • (2242nd in this Competition) • 3 years ago

0

amazing danzel! congrats! i tryied some models in the same line, could we share some knowledge? i did in python

danzel • (1797th in this Competition) • 3 years ago

0

What do you want to know?

daishu • (240th in this Competition) • 3 years ago

0

@springmanndaniel Can you post your code?

This comment has been deleted

denisond • 2 years ago 0

Hi @mjahrer ,

I have a few questions that I haven't been able to find answers elsewhere:

1. Looks like you used a DAE architecture of 221-1500-1500-1500-221. I thought the whole point of AE was to created some bottleneck, but the layers in the middle are larger. Does this bottleneck approach get thrown out for DAEs, and do you have a rule of thumb for this architecture other than trial and error?

You mentioned in your post:
`Two different topologies are used by myself. Deep stack, where the new features are the values of the activations on all hidden layers. Second, bottleneck, where one middle layer is used to grab the activations as new dataset. This DAE step usually blows the input dimensionality to 1k..10k range.`

Do you use both these topologies at once, or in this case would you just grab the 4500 features from the 3 middle layers? Also, is there any use to including the the denoised output as either new features or new samples?

1. Once you create these features, do you also include the original input features for your set of features that you put through the supervised nn? (so you end up with 4500 + 221 = 4721 features). Or do you just include the 4500 features?
2. Did you also try VAEs in this competition to create new samples, or has that proved a good technique for you in other competitions?
3. Was the input for the DAE mixed (sounds like it was exclusively one-hot encoded and RankGauss normalized numeric vars, please lmk if this is wrong), what loss function did you use for the DAE, and was this informed by the mixed data types?

Please let me know if you have answers to any of these. I realize the answers are often subjective but would greatly appreciated advice from someone who's applied this so well. Regardless, great post and thanks for all the information you've already shared.

Thanks

Michael Jahrer Topic Author • (1st in this Competition) • 2 years ago 1

Hi, thanks for your comments.

Yes the 221-1500-1500-1500-221 architecutre is not a bottleneck.
It is another kind of data coding by the help of denoising autoencoder training.
A different kind of code - it helped a little within the final blend.

Both topologies does a different data coding style.
It always depends what can work, it need to be tried out, always.

I remember trying to use the outputs too as features, but never helped the supervised model after. And yes you can also add original data, I guess was never trying this, but would be suprised if it helps.

Never followed the idea of VAE, my believe was better noise like swap noise add good variation to sample the feature space. In this competition I was lucky enough to apply this successfully.

Yes, categoricals as 1-hot, numerics are rank gauss normalized. This works really well since today. I found another schema for numericals which sometimes gives advantage over this rank gauss: thermometer rank coding. For one numeric feature I create eg. 100 binary features and use the thermometer code as representation (on a rank-equalized numeric scale).

Error for my DAE is: (y-y_target). I think this is called logloss error or crossentropy. Not MSE which is (y-y_target)^2.

Michael

DeepUnderstanding • 2 years ago 1

I like how even if the solution is 2 years old but people are still coming here for doubts and understanding.
So my doubt is that if I rank gauss my input numerical features, so in the output of my DAE model should I use the new normalised input or the original ones?
Thanks

DeepUnderstanding • 2 years ago 0

And why did you use `output is sigmoid` since the input may go less than 0 and more than 1?

Michael Jahrer Topic Author • (1st in this Competition) • 2 years ago 2

"gauss normalize": with this kind of transformation you do not loose information coded in numeric features, rank preserving and it seems better for nn's. Sometimes raw are better. You should have this in your "toolbox" for playing around.

As far as I remember, I havent tried normalization of neuron outputs (output of DAE). One should try this, I guess it doesnt help much.

With "output is sigmoid" I mean the problem is binary classification. The activation function for the supervised model after. Outputs

of DAE are linear.

Cheers!

denisond • a year ago

1

Hi Michael,

Thanks so much for the comments and apologies for the slow response, took a break from kaggling.

I am bit confused about your last point on the loss function for DAE. Doesn't log loss or BCE require targets be 0 or 1? I get how BCE is essentially just (y_target - y_hat), but if anything wouldn't we want to take absolute value of y_target-y_hat?

Just curious because I see a lot of talk about training autoencoders with MSE instead of BCE when inputs/outputs are not between 0 and 1, such as here: https://stackoverflow.com/questions/52441877/how-does-binary-cross-entropy-loss-work-on-autoencoders

In the end I guess it comes down to what works best, but curious about what loss function you used since it worked so well here. Am I correct in guessing it was something like abs(y_target-y_output) as opposed to regular log loss?

Thanks again so much for this great kernel and your response.

P.S. one follow up question: do you typically experiment with dummying out numeric variables like you mentioned above with thermometer coding? Haven't seen a lot of about when to attempt this wrt the number of levels in a variable.

Michael Jahrer Topic Author • (1st in this Competition) • a year ago

2

I use cross entropy, (y_target-y_prediction). This works usually best. But as you said it comes down what works best, you can try squared error, L1 error, CE error.

Thermometer coding (with equal ranked buckets) is a very robust code for representing numeric values to a nn. It always works. The bad thing is that you need 10..100 inputs to represent one num feature. And it can do worse than original numeric feature itself. Again here try yourself which normalization works best. CV will tell you.

denisond • a year ago

0

Thanks so much for your responses. The thermometer coding looks interesting, will definitely check it out.

I am still confused as to how you would calculate cross entropy for the DAE on numeric variables out of the range [0,1] (which I think should be the case for the rankGauss transformed vars? I believe they are roughly [-2,2]), did you end up scaling these so that they were in the range [0,1] but still gaussian? I also saw in previous comments that you said you trained it on MSE, so I am struggling a bit. Just training on MSE would make more sense, but also might have to consider the differences in scale between ohe and numeric vars, I think...

Hate to burden you with more questions, but I did run into one other comment I was confused about:

1. I saw above you wrote:

*Basically I removed *calc, added 1-hot to *cat features. Thats all I've done. No missing value replacement or something.*

I understand that ae are a great tool for learning a more general representation of our data. But how were you able to train a DAE with missing values as input? Is there some work around, or did you do basic imputation on cat and num vars before training the DAE?

Thanks again so much for this great kernel and your prompt responses 4 years later. We really appreciate it!

Mustafa Ünlü • 3 years ago

1

Intresting

Ahmed Fayed • 3 years ago

0

Interesting*

CPMP • (29th in this Competition) • 5 years ago

31

This is jaws dropping... Thanks for sharing so much details about your magic, it will take me time to digest your wisdom on DAEs. I keep seeing DL promoters say it can replace feature engineering, but few provide a convincing demonstration outside image processing. You just did it. Congrats for that, and of course for your undisputed lead on the LB from start to end. Thanks for the kind words too.

Ashish Anand • 3 years ago

1

Intresting

Bojan Tunguz • (374th in this Competition) • 5 years ago

19

Wow, very impressive work Michael. I feel like someone has sent us a solution from the future of machine learning. :)

I am also glad to see that I am not the only one who's not into feature engineering. It seems that there is still a place for me in the world of machine learning though. :)

Congratulations once again. It will take me a bit to digest your insights. I've already learned so much from this competition, and this is taking it to a whole new level. Hope to see even more of your interesting work in the future.

Giba • (53rd in this Competition) • 5 years ago

20

Congrats Michael for your solid victory! Your solution is very impressive and your description very detailed. I completely forgot to use denoising autoencoders here, last time I tried it was 4 years ago in "The Black Box Learning Challenge"

william • (739th in this Competition) • 5 years ago

17

Though there has been some discussions about RankGauss normalization. There isn't a correct and self-contained python version. Here is my Python solution based on some former comments.

```
import numpy as np
import pandas as pd
from scipy.special import erfinv
import matplotlib.pyplot as plt

def rank_gauss(x):
    # x is numpy vector
    N = x.shape[0]
    temp = x.argsort()
    rank_x = temp.argsort() / N
    rank_x -= rank_x.mean()
    rank_x *= 2 # rank_x.max(), rank_x.min() should be in (-1, 1)
    efi_x = erfinv(rank_x) # np.sqrt(2)*erfinv(rank_x)
    efi_x -= efi_x.mean()
    return efi_x

x = np.random.rand(500)
# histogram test: the histogram of rank_gauss should be gauss-liked and centered
pd.Series(x).hist()
plt.show()
pd.Series(rank_gauss(x)).hist()
plt.show()
```

This comment has been deleted

JustTesting • (1324th in this Competition) • 4 years ago

4

@Tom: such a self-contained python version of RankGauss was introduced last year to sklearn and is called "QuantileTransformer".

At least for continuous variables Michael's RankGauss transform can now be reproduced in python using `sklearn.preprocessing.QuantileTransformer()` function with `output_distribution` parameter set to 'normal' and `n_quantiles` set to 1/10 of the number of observations. It still fails for discrete distrubutions though, such as Poisson.

BTW. The second argsort() in your code is not needed, but multiplication by sqrt(2) is (to get unit variance).

Mallikharjuna Rao Manchem • 3 years ago

1

Good Job!!

CodeDevil • 3 years ago

1

Great explanation, thank you for sharing

Skinish • (332nd in this Competition) • 5 years ago

12

Is anyone interested in reproducing this in python? :)

Bojan Tunguz • (374th in this Competition) • 5 years ago

4

Oh I would love to see someone reverse engineer this!

Russ Wolfinger • (19th in this Competition) • 5 years ago

14

My main takeaway is that the two most important continuous predictors, car_13 and reg_03, are both noisy and smoothing them improves generalization in a neural net context. Michael's autoencoding approach appears to be the best way to tackle this here, along with providing a clean way to impute the huge chunk of missing values for reg_03 and picking up and smoothing deeper structure across the entire feature space (e.g. creating a nice embedding for car_11_cat, whose values might even be mislabeled in some cases). (If any organizers are listening, it would be very instructive now to learn the true meaning of these and other variables.) Note the #2 team's approach of using boosted trees to predict each feature from others is another potentially good way to go about this. This stuff is all pretty amenable to implementation in python and opens up a line of interesting research questions such as comparing swap noise to other approaches, how to perform feature extraction, and perhaps even a single end-to-end trainable architecture.

Bojan Tunguz • (374th in this Competition) • 5 years ago

^ 8

So Russ, my main takeaway is that you just volunteered for the job of implementing all of this in Python. ;)

Andy Harless • (21st in this Competition) • 5 years ago

^ 8

The Python implementation is also definitely something I would like to see. I'm not volunteering for the job, but I wouldn't rule out a minor role in a group effort.

Skinish • (332nd in this Competition) • 5 years ago

^ 1

that's the spirit!
PS: Have you shared your solution, Andy?

Bojan Tunguz • (374th in this Competition) • 5 years ago

^ 2

@Andy, OK, fine, I'll volunteer effort too.

Shabie Iqbal • (48th in this Competition) • 5 years ago

^ 1

Count me in too!

Edit: Though I am not sure how useful I can be.

Russ Wolfinger • (19th in this Competition) • 5 years ago

^ 6

@Bojan haha suggest the work → get the work. If anyone wants to take a crack at a new kernel I'm sure you would get some action on it and I'll be the first to upvote it. Besides that and any students who want to take this on as a degree project, I guess maybe the most practical plan is to try and reassemble our stellar team of kernel writers for the next similar competition. This comp might take the cake for the best set of kernels ever, and I do not think the primary ones overfit much if at all. Our stack gave about 50% weight to them and our other half had some similar feature engineering to what has already been described by Little Boat + FFMs. I was speculating Michael was going to roll out some crazy good FFM but the DAE approach makes much more sense.

NxGTR • (347th in this Competition) • 5 years ago

^ 4

I love when people start to discuss implementing stuff in proper languages.
Thanks for not thinking about doing it in R.

Bojan Tunguz • (374th in this Competition) • 5 years ago

^ 8

I have no idea what R you talking about.

Shabie Iqbal • (48th in this Competition) • 5 years ago

^ 2

LOL! Now it makes sense. R was created only for this joke :P

CPMP • (29th in this Competition) • 5 years ago

^ 0

> I love when people start to discuss implementing stuff in proper languages.
> Thanks for not thinking about doing it in R.

Is R a language?

CPMP • (29th in this Competition) • 5 years ago

^ 0

> This comp might take the cake for the best set of kernels ever, and I do not think the primary ones overfit much if at all.

Then why aren't these among the top entries in private LB? That you reused some of them in your blend is not a proof that they aren't overfiting...

Mileta • (2511th in this Competition) • 5 years ago

5

Starting points :

https://charleshsliao.wordpress.com/2017/06/26/denoise-with-auto-encoder-of-h2o-in-python-for-mnist/

https://cran.r-project.org/web/packages/SAENET/SAENET.pdf

https://github.com/h2oai/h2o-3/blob/master/h2o-r/tests/testdir_algos/deeplearning/runit_deeplearning_autoencoder_large.R

mlhammer2logit2quit • (1558th in this Competition) • 5 years ago

0

"swap noise" from samples within common "predicate blocks" could be interesting.

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

4

thanks, I tried this with #4 model (you see colGroups=1 in unsupervised params). This was the idea to swap columns with same meaning (categoric + all corresponding 1-hot categories). No success...

mrxnew • (288th in this Competition) • 5 years ago

4

@Michael Did you try training the DAE network and the neural network as one big model ?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

1

no. But I put it on the todo list for future experiments ..

Russ Wolfinger • (19th in this Competition) • 5 years ago

5

@CPMP certainly not definitive proof but evidence that it is possible to do well on the private lb while assigning significant weight to kernels. We locally cross-validated each component to verify and found nearly all to be decent and amenable to tuning, although did find Scirpus' wild and exotic bigGP unweildy, even after jacking it up with Andy's noise trick (really need to compute true out-of-folds for it). It would be interesting to summarize to what proportion kernels are used in the top finishers; looks like for both you and Michael that would be zero! For Kagglers this set of kernels is a great collection from which to learn, along of course now with a shining wonderful early-Christmas example of the potential utility of denoising autoencoders.

CPMP • (29th in this Competition) • 5 years ago

0

> @CPMP certainly not definitive proof but evidence that it is possible to do well on the private lb while assigning significant weight to kernels. We locally cross-validated each component to verify and found nearly all to be decent and amenable to tuning

Fair enough, my point was more about reusing them as is without any further check. You clearly didn't had this myopic approach, nor did others above me. Thanks for the answer.

Kerem Turgutlu • (1088th in this Competition) • 5 years ago

3

I am trying to implement here is my starting point, I am actually not very familiar with OOP and hope that I didn't misinterpret any part of the solution since it would cause going from winning to losing :) Hope to get some tips as well, thanks !
https://github.com/KeremTurgutlu/deeplearning/blob/master/DAE.py

YaGana Sheriff-Hussaini • (62nd in this Competition) • 5 years ago

1

@Russ W, count me in too for porting @Michael Jahrer's solution to python group.

Hail the Snail • (3525th in this Competition) • 5 years ago

0

Does anyone has any idea how do you do 1-hot labeling with the columns that has -1 values?

Roberto Spadim • (2242nd in this Competition) • 4 years ago

3

some work here: https://www.kaggle.com/rspadim/simple-denoise-autoencoder-with-keras

Μαριος Μιχαηλιδης KazAnova • (9th in this Competition) • 4 years ago

6

This must be the most popular (in-competition) thread I have ever seen in kaggle. It is amazing that after so many months discussion is still on :)

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

6

representation learning is the future :) ^^

Bojan Tunguz • (374th in this Competition) • 4 years ago 0

> **Michael Jahrer wrote**
>
> representation learning is the future :) ^^
> That's **exactly** what I said in my comment when I first read about this solution. :)

Jiazhen Xi • (155th in this Competition) • 4 years ago 3

You mean ***Generative Adversarial Denoising Autoencoder*** ? :)

Bojan Tunguz • (374th in this Competition) • 4 years ago 2

Exactly :)

Bojan Tunguz • (374th in this Competition) • 4 years ago 1

OK, this is going to sound crazy, but has anyone ever thought about using some kind of decision-tree approach to autoencoders? Asking for a friend ...

Oscar Takeshita • (69th in this Competition) • 4 years ago 0

@Bojan. Are you already done with step one?

Bojan Tunguz • (374th in this Competition) • 4 years ago 0

Step one?

Oscar Takeshita • (69th in this Competition) • 4 years ago 0

Trying first with a neural network :) One thing I'm not sure how to handle with decision trees is multi-dimensional output. Would we need something like that to make it work?

Bojan Tunguz • (374th in this Competition) • 4 years ago 0

No. Yes. I want to try decision trees for a little project I'm working on.

Oscar Takeshita • (69th in this Competition) • 4 years ago 1

Maybe we should try to read something like **this**

YaGana Sheriff-Hussaini • (62nd in this Competition) • 4 years ago 1

Thanks @Oscar Takeshita for this paper. I think I will give this a try on another project.

> This must be the most popular (in-competition) thread I have ever seen in kaggle.

@KazAnova, that says a lot about @Michael Jahrer's solution here as well as Porto Seguro.

Bhupinder • (1847th in this Competition) • 4 years ago 2

Yup one of them. And I keep coming back to this post and try to learn as much as I can. Michael I have one doubt. When you are training an autoencoder, how to decide the epochs? In the log, i don't see any validation rmse. So how you decided to use 1000 epochs? Is there any concept of overfitting in autoencoders?
And again thanks :)

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago 5

with denoising setup (swapNoise) you can never learn down to zero training error. there is a lower error bound, hence no regularization needed. train down as best as you can is my advice. But there is also some black art involved, for example lower training error does not necessarily mean better results on supervised task afterwards. Many unknowns .. but at least I managed for some datasets to show that DAE can do something useful.

Bhupinder • (1847th in this Competition) • 4 years ago

0

Thanks Michael.

Bhupinder • (1847th in this Competition) • 4 years ago

2

Btw found this paper on swap noise if someone is looking for more details on that. They also talk about using it in images as replacement for dropout. https://arxiv.org/pdf/1801.07316.pdf
(Michael you are mentioned in the paper :))

Oscar Takeshita • (69th in this Competition) • 4 years ago

0

@kuroashi Thanks for the paper. I wonder if anyone has successfully reproduced Michaels denoising results.

Roberto Spadim • (2242nd in this Competition) • 4 years ago

0

interesting, i'm considering if a generator is better than a activation function in terms of gpu use

maybe instead of random sampling from dataset, could sample from a empirical distribuition, nice to see this paper https://arxiv.org/pdf/1801.07316.pdf, i didn't found swapnoise, but i didn't search about bootstrap, thanks!

Roberto Spadim • (2242nd in this Competition) • 4 years ago

2

@Tunguz, check this https://github.com/xiaozhouwang/kaggle-porto-seguro/blob/master/code/fea_eng0.py

maybe could guide to something near to tree autoencoder

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

3

thanks for pointing to this paper, very interesting. Why they are using the word "Hybrid Bootstrap" for that? Why hybrid? They sample from another training point, this is exactly what I was doing here.
Nice to see they use it as a replacement for dropout :) I also tried exactly this in the hidden layers (as replacement for dropout) a year ago or so with no success. Anyways. funny.
Again, for generating artificial noise samples for a denoising autoencoder swapNoise (or hybrid bootstrap) are good choices for csv/table data. But there must be better ways to generate artificial samples (GANs).
For image data all kind of natural distortions (shear, affine, scale, rotate, flip..) are better ones.

Roberto Spadim • (2242nd in this Competition) • 4 years ago

1

if i`m not wrong, if you execute any grandient based tree, you have many many 'layers', the output of each 'layers' could be used as 'features', but tree probably isn't the best thing to create noise since the output is a linear transformation of input, probably tree can make something like if (x1 > -inf): y=x1*.9999999999999

in other words it don't 'create features', i didn't tested but maybe it's something to check, instead of decision tree and forest, use gradient based trees. the problem now is what input/output to set, x=x or something like x[0-90% of features] = x[90%-100% of features] (predict others features instead of predict same features)

i used last url as reference, and did some work here, must check if this is usefull or not

https://www.kaggle.com/rspadim/autoencoder-with-xgb

well i tried denoise with xgb, let's see if it works

code here: https://www.kaggle.com/rspadim/denoiseautoencoder-with-xgb

i don't know if it's ok, but a good start of code, i think... good luck guys

BrainFuck • 4 years ago

1

Hi,this is great! I am very interested in "Representation Learning", do you have any related material to learn more about this tech? Thank you very much!

pooh • 4 years ago

0

I think courses on deep learning cover the representation learning techniques discussed here.
Currently i am enrolled in Udacity deep-learning Nanodegree and they are teaching all these topics. Having said that it is not necessary that you should enroll in that course. Other courses that are taught by ods.ai and fast.ai also cover those topics.

Sobrino Mario • 4 years ago

2

I was looking for the same thing & found this paper:
https://arxiv.org/pdf/1206.5538.pdf

It's a long read if you want to know all the details, but section 2 & 3 cover what I wanted to know.

Muhammed Muqtar Fasasi • 4 years ago

1

Seems like I got a lot to cover...
Still struggling with the terminologies

YaGana Sheriff-Hussaini • (62nd in this Competition) • 4 years ago

2

Muhammed @shiwayz, take your time and learn well. I recommend you take the following free foundational Machine Learning course by Andrew Ng.

https://www.coursera.org/learn/machine-learning?utm_source=gg&utm_medium=sem&campaignid=685340575&adgroupid=32639001341&device=c&keyword=coursera%20introduction%20to%20machine%20learning&matchtype=b&network=g&devicemodel=&adpostion=1t1&creativeid=243289762778&hide_mobile_promo&gclid=EAIaIQobChMIyMaVwoXn4AIVAtbACh2n0A5WEAAYASAAEgLk6_D_BwE

Muhammed Muqtar Fasasi • 4 years ago

0

Thanks man.
I've already enrolled even...
But haven't started yet
I should start ASAP then
Once again, thanks man.

Little Boat • (2nd in this Competition) • 5 years ago

7

Oh wow. This is absolutely beautiful! I really love doing things in a fairly simple way and your solution is like, faaaaaaar more elegant than mine. Well deserved!! I have to admit that I don't have much faith on the accuracy of neural networks on tabular data, or, unsupervised learning for feature engineering in general. But I will definitely think of them for future competitions! Thanks Michael!! And congrats again!

Just one quick question, how much did "swap noise" help in your models?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

9

thanks man. Without swapNoise DAE would not work that well. But this is also not optimal as well

Little Boat • (2nd in this Competition) • 5 years ago

4

Got it. I have never thought of swapping part of features to create some noise (only things like some variants of gaussian noise, etc. which rarely worked for me). But now it seems so natural to me! Learnt something great from you again!

Still remember the excitement I got while reading your netflix solution. Keep inspiring us please!!! :)

Abhishek Thakur • (2017th in this Competition) • 5 years ago

7

Thank Michael!! Will you be releasing the code?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

9

Hi, short answer no. Maybe in future. Exactly same nn code is used here at operasolutions in our ml module. Also who will use a hacky command line tool ? 99% use python packages

jmbull • 5 years ago

0

> who will use a hacky command line tool ?
> currently there are just too much python/R packages that bust my head

I love your humility :)

Congratulations on the definitive win! I love reading write-ups like this because it takes me a week to unpack all of the terminology that's new to me. This contest has caused me to prioritize learning more about NNs!

olivier • (33rd in this Competition) • 5 years ago

8

Congatulations Michael Jahrer and thanks for sharing your solution in that much detail. All this in C++/CUDA is really impressive ;-) I'll remember rankGauss and denoising auto encoder for future competition (but I'll have to invest in a good enough GPU ;))

KALE • (27th in this Competition) • 5 years ago

Very impressive! The knowledge you shared here is a real treasure for Kaggle community!

8

Simbad • 4 years ago

Great post ! Thanks for sharing

1

Evgenii Kononenko • (951st in this Competition) • 4 years ago

Great job!

1

Woonsung Baek • 4 years ago

Thanks for sharing!

1

william • (739th in this Competition) • 5 years ago

One short question, do you use batch normalization layers in your DAE nn or supervised nn?

3

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

no, all plain SGD

1

william • (739th in this Competition) • 5 years ago

Thank you, very timely answer!

0

Ee Kin Chin • 4 years ago

If you use rankgauss method mentioned, subtracting the mean in the end the output will be features that will be lesser than 0, using sigmoid means it can't reconstruct values below 0, how will this work?

1

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

yes, transformed values are in -3..+3 range after normalization. In DAE I use linear output (no activation function), they can approximate any range.

1

Ee Kin Chin • 4 years ago

Thanks for the clarification Michael!
Also I noticed you used auto encoders that has larger dimension in the middle layers compared to the input dimensions, could you help to explain what's the difference of using larger dimensions vs using smaller dimensions in the middle layers of DAE?

0

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

with more neurons in hidden the model has more power to learn the denoising function

1

Ee Kin Chin • 4 years ago

Thanks for another important clarification! Final question, do you think the training RMSE score for DAE you achieved is representative of the amazing score you got in the LB or do you think the 1000 epochs training with a generally downwards error trend after every epoch is more important? I see you achieved 0.05 RMSE levels in training but for my dataset I am getting 0.200 range of RMSE and slowly going down to 0.19 RMSE range. Do you think I should stop at 1 or 2k epochs or only stop at a similar training RMSE you achieved here(Assuming that it can even reach it)? What's your take on this?

0

Oscar Takeshita • (69th in this Competition) • 4 years ago

@Chin Ee Kin: Looking at @Michal Jahrer's log file, the loss value starts at 0.121301 at the end of the first epoch and finishes at 0.0533052. In my experiments, the loss values get more similar to his if I use MSE instead of RMSE. I think the difference you are seeing might due to this. I wonder if @Michael Jahrer's is actually computing MSE but labeling the value as tRMSE in the log file.

0

AnkurSaikia • 5 years ago

I have a confusion regarding the RankGauss Normalization step. Please follow through my step and correct me if I am wrong.

3

- Say i have a feature column X

- Say I have a feature column X
- Then for all unique values in X, I sort the values and assign a linspace'ed rank in range (0,1)
- Then I apply Erfinv on the ranks
- Then for each value in X, we assign the result of previous step(Erfinv) for that value
- Now we calculate the mean of new transformed X, and substract it from each row in the column X

My Python Code:

```
import numpy as np
from scipy.special import erfinv

def RankGauss(X):

    # create the list of unique values in X and sort them
    sorted_items = sorted(set(X))

    # Rank each of the values in linspace (0,1)
    sorted_ranks = {}
    rank = 0

    for i in sorted_items:
        sorted_ranks[i] = rank/len(sorted_items)
        rank+=1

    # Create a list of normalized X values
    normalized_X = []
    for i in X:
        normalized_X.append(sorted_ranks[i])

    # Apply erfinv on the ranks
    normalized_X = erfinv(np.array(normalized_X))

    # subtract the means from the results
    normalized_X = normalized_X - np.mean(normalized_X)

    return normalized_X
```

tiopon • 5 years ago

0

is these steps confirmed yet? anyone who can share links about "RankGauss" is also highy appreciative.

william • (739th in this Competition) • 5 years ago

1

This code doesn't work correctly. I have done the histogram test, the histogram of the returned RankGauss(X) isn't gaussian and centered. You can see my solution in this comment

Kerem Turgutlu • (1088th in this Competition) • 5 years ago

3

Hi @Michael Jahrer congratulations and thanks for sharing your amazing invention. I have a question in normalization part, you said that we need to rank and then apply linspace between 0 and 1 then apply erfinv. But what I've found online for this transformation says we need to first have an interval like (-1, 1) then to apply y = sqrt(2)*erfinv(x) in order to get gaussian distribution. I am a bit confused about this. Once again thanks !

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

0

hi, thanks. Yes I think you are right, linspace should be -1..+1. Just look at the histogram after the trafo. When its gauss-shaped and centered you've done it right.

Kerem Turgutlu • (1088th in this Competition) • 5 years ago

0

Thanks!

Vineet Kumar Singh • 5 years ago

1

Trying to implement rankGauss in python, here are my steps

1. Get the index of the series
2. sort the series
3. standardize the series between -1 and 1
4. apply erfinv to the standardized series
5. create a new series using the index

Am i missing something ??

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

6

I subtract mean afterwards. And do not touch 1/0 (binary columns). The basic idea of this "RankGauss" was to apply rank trafo and them shape them like gaussians. Thats the basic idea. You can try your own variation of this.

Ravi Teja Gutta • (804th in this Competition) • 5 years ago

3

Certainly you deserve to win with this kind of work. This is going on for some time where we let model do Feature Engineering for us. It was already proved in vision and language and to some extent in tabular data. This shows how neural nets are so expressive in their modeling capabilities. And on the long run I can see a trend in kaggle and ML practitioners around the world adopt to this automatic feature engineering magic.

Rafael Carvalho • 4 years ago

1

Hey Michael, thanks for sharing your job!

I just have a question about the batch variables corruption. As you answered @Kerem Turgutlu:

> Hi Michael, did you corrupt variables at each batch on the fly with shuffle or before training stage for only once ?
>
> at each batch. Each batch samples new noise from the complete dataset. Complete dataset is train+test features. Copy the features before to use as target. Targets should be the clean uncorrupted features.

Is there any difference between sampling new noise from the complete dataset at each batch on the fly and corrupting the variables (shuffling) once at each epoch from the complete dataset then using it to generate the batches? In my conception both cases will generate brand new batches every epoch.

Thanks!

Kerem Turgutlu • (1088th in this Competition) • 4 years ago

3

Conceptually I think they are the same since you will introduce probability of corruption. But doing it at each batch might be easier in terms of frameworks like PyTorch where you only need to define a transform inside your dataset class. General idea if I am mistaken is same as data augmentation in image classification, where you want to have infinite number of combinations of samples to better generalize. But let's wait :)

Rafael Carvalho • 4 years ago

1

I think you're correct :) My doubt was if it was conceptually different or was just a "technical" issue, your answer leads me to think that it's just a matter how you implement it.

daishu • (240th in this Competition) • 4 years ago

1

Please forgive my poor English. Does the number of hidden layer neurons in the DAE greatly influence the effect of representation. Such as 221l-500-500-500-221l. And, is all the rows doing SwapNoise?

Thanks!

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

2

yes, width of hidden layers determine model power. More is usually better, at least 1000 neurons I would say to fully occupy GPU (batchsize 128). More should not hurt when trained with swap noise (=strong regularizer, error can never be 0).

daishu • (240th in this Competition) • 4 years ago

0

Thanks for your reply. I haven't been able to achieve your magic. Maybe I need a bigger nn.

daishu • (240th in this Competition) • 4 years ago

2

I am wondering that whether you input SwapNoiseData or initial Data when you finally extract new features from hidden layers?

Skyfacon • (1424th in this Competition) • 4 years ago

0

你有复现他的自编码器提取特征方法吗？能一起交流下吗？

daishu • (240th in this Competition) • 4 years ago

3

If I achieve Michael's score, I will post my Python code.

Boian Tunguz • (374th in this Competition) • 4 years ago

4

> **袋鼠 wrote**
>
> If I achieve Michael's score, I will post my Python code.
> Even if you don't achieve it but come close, please post your code. It would still be very useful.

This comment has been deleted

YaGana Sheriff-Hussaini • (62nd in this Competition) • 4 years ago 0

Great job and thanks @Kangaroo, I would love to see your code.

daishu • (240th in this Competition) • 4 years ago 1

Thanks for your encouragement, I have a long distance from magic.

Bojan Tunguz • (374th in this Competition) • 4 years ago 2

Start a GitHub repository. Maybe others would join in. This doesn't have to be a solitary effort.

Jacek Poplawski • (20th in this Competition) • 4 years ago 2

Hello Michael, I am trying to use your tips to increase score in another competition.

I have generated simple dataset I filled nan with mean (in Porto Seguro there was no nan but -1), I have applied MinMaxScaler (no RankGauss).

I have implemented swap noise - take some features from different row in same batch.

Target was not used as feature, so it is not in input or output layer.

I have 241 features.

Then I tried multiple architectures for autoencoder:

- 241-241-241
- 241-482-241
- 241-482-482-241
- 241-1000-1000-241
- 241-1000-1000-1000-241

With swapRatio 0.15 I can't beat mse 0.0061.

What was your value of loss after 1000 epochs?

(I tried with swapratio 0 and I was able to reach loss 0)

I then tried to use generated values from hidden layers.

With and without original dataset. With both LGB and NN.
I don't see better score on my local validation.

Maybe I am doing something wrong, I suspect loss of autoencoder should be smaller, right?

Or do you think RankGauss is so important? Why?

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago 5

yea you figured it out - there are so many unknowns in this unsupervised phase. Years ago I also tried something like this and I got lucky to found a setting where error improves locally and I followed this path harder, this is what this solution here is about. There are many details, begins with data normalization to topology, learn params, etc.. The mse you mentioned with .0061 cannot be compared with my log output coz input data is totally different, hence I cannot say weather this is good or not. Ultimate test is using the generated features to the supervised learner and see some gains. Recently I got an invite to join Bojans team @ "home-credit-default-risk", maybe dae+nn helps also there. I'm not too optimistic because it has only a small testset.

DeePBluE • 4 years ago 1

I had the same case as you. Almost same network structures, also used swap noise, the only difference is I didn't use RankGauss. However the score was very low. I wonder if there are some important parts that we missed.

Jacek Poplawski • (20th in this Competition) • 4 years ago

0

Michael what are your experiences with Home Credit? :)

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

3

:) rankgauss+dae+nn does not work as well as plain lgbm. Cannot breach .79 barrier on CV :(
still trying .. but this is the regular behavior. neural nets are behind boosted trees on csv datasets. I saw this so many times. One some datasets (like here on porto) it works. Especially when nTest > nTrain and train+test comes from same distribution, unsupervised approach can learn better representations, "sometimes" :)

Jacek Poplawski • (20th in this Competition) • 4 years ago

1

I can't beat 0.784 on NN :) do you use simple architecture like 1000-1000 or 200-10 or very deep one?

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

1

i used 10000-10000-10000. 20% swapnoise. Data normalization is the critical part. Which numeric preprocessing have you done?

Jacek Poplawski • (20th in this Competition) • 4 years ago

1

I have read your post many times, and I implemented input-15000-1000-1000-1000-15000-output with swaprate 0.20, I measure accuracy and it is 43% after training, then I have used 1000-1000-1000 values as an input for NN then this NN is able to get 0.784 on my CV, I have tried many architectures but no matter is it 2000-2000 or 200-20 it always finish around 0.78.

Fedor • (1899th in this Competition) • 4 years ago

0

I ran into the same value and can not go higher ((How many input features have you got?

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

0

between 300 and 700, depending on which dataset

DeePBluE • 4 years ago

2

Hey, I wonder if those unsupervised features are also uesful for tree models like lightbgm? Or only have magical effect on NN models. I noticed that you didn't train any lightgbm model using unsupervised featues in stage2. Thanks!

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

1

proper question. Yes I tried that (dae+lgbm) but gave only minor improvement in blend. As far as I remember it dropped LB score, so I did not experiment further.

tomcwalker • (386th in this Competition) • 5 years ago

4

Very nice writeup. Great to see a creative application of NNs beating down the stacks upon stacks upon stacks of varied models. Autoencoders were on my list of things to try that I didn't get around to this time.

I predict a huge upsurge in autoencoder related kernels.

Congratulations and thanks again!

CPMP • (29th in this Competition) • 5 years ago

1

I was thinking the same! Michael is reviving something that is considered a thing of the past in deep learning research community!

Shabie Iqbal • (48th in this Competition) • 5 years ago

4

Well I am the farthest thing from an expert but comparing this to the other solutions presented so far, yours takes it to another level. I know Kaggle will soon be coming up with another level beyond Grandmaster for people like you.

Congratulations!

Seeker • (668th in this Competition) • 5 years ago

4

Thanks a lot Micheal !! Well deserved first position, I am amazed by the fact that you have used c++ for coding.

Lesaffrea • (1405th in this Competition) • 5 years ago

1

Inverse error function

erfinv ← function (x) qnorm((1 + x)/2)/sqrt(2)

RankGauss ←function(torank){

```
    if( !is.data.frame(torank) &amp; !is.matrix(torank))  { stop(" The input data must be one data frame or
matrix")}

    allrank  <-apply(torank,2,rank,ties.method="min")

    allsort  <-torank[!duplicated(torank),]

    allsort  <-as.data.frame(sapply(torank, sort))

    rowrank  <-rowSums(allsort)/ncol(torank)

    rowrank  <-data.frame(allsort,

                        Value=rowrank)

    rowrank$RankScale  <-seq(from=-.9999, to=.9999, length.out = nrow(rowrank))

    # We use the inverse sigmoid

    rowrank$RankDist  <-with(rowrank,erfinv(RankScale))

    back <-apply(allrank, c(1,2), function(onerank){rowrank$RankDist[onerank]})  #Very slow

    back <-as.data.frame(back)

    names(back) <-names(torank)

    return(back)
```

}

william • (739th in this Competition) • 5 years ago

0

Can you re-edit this R code episode to a readable version, i.e., multiple lines with indent.

Lesaffrea • (1405th in this Competition) • 5 years ago

0

Reformat :) Sorry

william • (739th in this Competition) • 5 years ago

1

The proportion of Positive/Negative is very unbalanced. Haven't you done anything of upsampling before feed the train set into the neural network?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

0

in my models here no. Maybe there is some potential, I tried upsampling in a single nn on raw data and it improved to non-upsampling version. But on raw data I could not get better than 0.286 on CV or so, too low to contribute to the blend.

william • (739th in this Competition) • 5 years ago

0

Thank you. And you mean that you dont use the upsampling on the DAE features either?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

0

no, everything on the 595212 train samples.

Ahmad Obiedat • 5 years ago

1

Oh I would love to see someone reverse engineer this!

Francisco Azuaje • 5 years ago

1

As others pointed out: good to see the utility of autoencoders. However, I wonder if you could provide more details about the selection of hyper-parameters. Nice post, thanks for sharing

parameters. Nice post, thanks for sharing.

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago 4

thanks. I started with some params that worked on common problems like MNIST or other competitions. Like swapNoise=0.15 is a good start. Or 1-3 layers of 1000 units for the supervised nn. Learning rate for DAE is good when its near to divergence (as large as possible). DAE topology for deep stack 2-5 hidden layers is common. Sometimes memory limits the number of hidden units. Dropout in supervised nn is always a good idea. Values 0.1 .. 0.9 can be tried. lRate in supervised nn should be set that the net learns at least 100 epochs or so until it overfits.

daishu • (240th in this Competition) • 4 years ago 0

Hi, Michael. I am trying to reproduce your magic for a long time. I can't use SGD with small learning rate to converge. When I use a big learning rate, the result will be unstable.

I wonder why you can use 0.0001 learning rate fit so quickly in supervised learning . Can you kindly post the range of gradients and weights ?

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago 1

thanks, my lRate is a raw value. This means without dividing by batchsize of whatever. Try my lRate/128 and use batchsize=128

C_C • (68th in this Competition) • 5 years ago 1

Great job! Just a question，whether features set f0 needs to be normalized before DAE; or just normalize the hidden layer of DAE before nn models?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago 3

pipeline is: raw→normalization(=RankGauss)→DAE→nn
no normalization after DAE.

Daniel Möller • (3595th in this Competition) • 4 years ago 0

Pardon me if this is tooo newbie from me.... but....

If you're discarding all "calc" vars and inputting only one-hot encoded data, what is the role of normalization?

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago 1

"calc" feats hurt the generalization of the models here, so better to reject them.
one-hot is the more natural way to me to handle categoric data. For example cat=1 can be totally different to cat=2, so when I have 1-hot encoding these are different features, hence independent.
But all this feature generation / normalization (needed in nn) is still black art

Perfect Is Shit • (1749th in this Competition) • 5 years ago 1

Thanks and congratulation, I just have one question here, you put the lb score for lgb in the figure, does this mean wit the feature engineering you done and a fine-tuned lgb model, you can get 0.29?
P.S. I just want to know is there a way that one model can get a good score?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago 2

yes. 0.291 private score with lightgbm with 8[min] total train time. I think that's the best for a simple/quick model.

LongYin/杰少 • (661st in this Competition) • 5 years ago 1

I have a question, you say:
“train+test features to build the DAE. The larger the testset, the better”.
I think you take advantage of the test set which is usually not accessible. So, I think if you only do DAE on training data and then make predictions on test data, the score may drop a lot. Right?

liuyongqi • (120th in this Competition) • 5 years ago 0

no...everyone got the same dataset, you can use it in any way.

TensorFrozen • (212th in this Competition) • 5 years ago 1

I like this sentence "I never remove features"! Feature engineering is really painful.

DavidSutton • (479th in this Competition) • 5 years ago

1

Thank you for this explanation, Michael. This is inspiring work, and it's fantastic that you have shared your insights here: I'm looking forward to experimenting with DAE myself in the future. You said that you included the test data features when training the DAE. I was wondering if you have an insight into how well this approach generalizes to completely unseen test data?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

1

I've done a run to train the unsupervised DAE step without test features. Its described in one paragraph in my writeup. Here I run my framework in prediction mode like I never saw the testset before. Result got worse, but not much.

Vineet Kumar Singh • 5 years ago

1

I am trying to reimplement this solution in python.
one question about removing *calc features and one hot encoding *cat features. i am getting final 220 features..not sure what i am missing.

total features = 59

*calc features = 20

*cat features = 14

*bin features = 11

one_hot_encode on (*cat and *bin) gives me 206 features. so that sums up total features to
59 - 20 - 14 -11 + 206 = 220

What am i missing anyone ??

Kerem Turgutlu • (1088th in this Competition) • 5 years ago

2

Do the feature engineering mentioned in desciptions. Create ohe for cats but also keep the originals. You should get 221.

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

3

yes, see:

37 2..38 no *calc

5 ps_ind_02_cat

3 ps_ind_04_cat

8 ps_ind_05_cat

13 ps_car_01_cat

3 ps_car_02_cat

3 ps_car_03_cat

10 ps_car_04_cat

3 ps_car_05_cat

18 ps_car_06_cat

3 ps_car_07_cat

2 ps_car_08_cat

6 ps_car_09_cat

3 ps_car_10_cat

104 ps_car_11_cat

---

221 features

Bozza • (993rd in this Competition) • 5 years ago

1

Hey Michael, all,

Any specific reason why you didn't drop the original *_cat variables after you one-hot-encoded them? Isn't that redundant information? I thought that redundant information (like linear combination of variables) is never good both from a speed and an

accuracy point of view.

Thank you

ps: great solution!

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

1

I tried to drop, but CV got worse. So I kept the originals

william • (739th in this Competition) • 5 years ago

1

Amazing! You are the Michael Jordan & Michael Jackson in Kaggle!

robotcator • (43rd in this Competition) • 5 years ago

1

Congratulations and thank you for sharing. I have a dumb question cause I am new to nn. All your nn models are trained on DAE's hidden activations. How to determine the representation is good enough for training in nn model ? Thanks.

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

2

very good question. I don't have an answer. Trial and error. You can look at reconstruction rmse during training of DAE when having fix noise level. But its only an indicator. I've tried very deep and wide DAE with better rmse, but results in worse supervised score.

robotcator • (43rd in this Competition) • 5 years ago

0

Okay, got it. Thanks.

Tilii • (38th in this Competition) • 5 years ago

1

@Michael Jahrer Your solution is so deserving of prize and praise. Big props for your old school approach. Also, for keeping your nose ahead throughout the competition, as there were several talented teams at your heels.

Peter Hurford • (1403rd in this Competition) • 5 years ago

1

Well I think I know my two biggest mistakes:

1.) My model was far too complex (I had ~10 XGBs, ~10 LGBs, a few LRs, and a few shallow NNs combined by a higher-level LGB) and I guess I overfit. It looks like a linear regression combining them would have performed better and potentially got me a silver medal, but instead all I got was this learning experience.

2.) I don't know how to build NNs yet (though I did get decent results with `MLPClassifier` from Sklearn).

Jacek Poplawski • (20th in this Competition) • 5 years ago

1

this is awesome and very unexpected! now I have good excuse to focus on autoencoders :)

hope to see the code too :)

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

3

yea .. I am thinking of it. NVIDIA may donate a new GPU to open-source projects .. a V100 would be nice :D since I can switch to GPU16 2-byte floats (have full support coded for that) and the volta provide >100TFLOPs. gamer cards like the 1080 suck with half-precision floats.

Russ Wolfinger • (19th in this Competition) • 5 years ago

1

Awesome sauce! Thank you Michael for sharing and congratulations on a dominating performance.

makio323 • (77th in this Competition) • 5 years ago

1

Congratulation for the decisive winning and thank you very much for sharing the great details of your wining solution and the deep insights on ML! This is my textbook for the next competition.

qianqian • (2nd in this Competition) • 5 years ago

1

Absolutely brilliant! Congratulations and how come you find autoencoder works on this task?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

12

I tried DAE for years, on every attempt I found some minor improvements. They worked in otto as well but we didnt publish it.

KALE • (27th in this Competition) • 5 years ago

0

Otto was three years ago...

Morty • (1829th in this Competition) • 5 years ago

2

Thank you for sharing this very interesting solution! I am kind of kicking myself for not having used DEA in my submission, since I've been using them extensively in other projects. Seems like it really did the trick to avoid over-fitting for the DNNs.

I am curious about the autoencoders you applied. I've never seen this kind of design without a bottleneck before. Except for the heavy dropout rate it does not seem to do the usual encoding-decoding. Could you explain why you did it like this, and did you try a bottleneck design? Or did I misunderstand your layout here since you called 221-1500r-1500r-1500r-221l a bottleneck?

I am not sure if I could track down the inspiration of the InputSwap Noise. Was this the inspiration?:
https://lvdmaaten.github.io/publications/papers/ICML_2013.pdf (Learning with Marginalized Corrupted features by L van der Maaten - 2013)

Thanks a lot!

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

2

Thanks too. Denoising autoencoders does not need to be symmetric or have a bottleneck layer. Their aim is to reconstruct a "noisy" version of the sample. I figured out that a deep stack of same layer size like input-N-N-N-output does a good job. Thanks for the paper link, I will read it when time let me do. No swap noise was just implemented as I think about the "noise generator". In tabular data you have mixed colums like binary, categoric integers and numeric floats. With this swapNoise one can sample noise values from the same distribution column-wise. 10..15% swapNoise is a good start value.

qbit271 • (248th in this Competition) • 5 years ago

2

Thanks for sharing and highlighting the value of unsupervised learning!

Jacek Poplawski • (20th in this Competition) • 5 years ago

2

Michael, one more question, I think nobody discussed that before.
You said 32GB RAM and 1080Ti.
What about OS? Are you on Linux, Windows, something else?
I am asking because 1) memory management 2) CUDA

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

11

ubuntu 16.04 what else?

Bojan Tunguz • (374th in this Competition) • 5 years ago

4

You disappoint me. Here I thought you were able to pull all of this off on a Raspberry Pi. :P

Victor Paslay • (303rd in this Competition) • 5 years ago

6

I'm totally sure he managed to run Ubuntu Unity on Raspberry PI.

CPMP • (29th in this Competition) • 5 years ago

2

This post is probably the most popular post ever on Kaggle.

Peter Klauke • (137th in this Competition) • 5 years ago

1

Not in terms of the number of upvotes: https://www.kaggle.com/c/passenger-screening-algorithm-challenge/discussion/35118

CPMP • (29th in this Competition) • 5 years ago

1

Thanks for correcting me.

Andy Harless • (21st in this Competition) • 5 years ago

1

Anyhow this is the first thread that got me to install Firefox on my phone so as to read a properly threaded rendering.

Μαριος Μιχαηλιδης KazAnova • (9th in this Competition) • 5 years ago ▲ 2

Well done - impressive result!

liuyongqi • (120th in this Competition) • 5 years ago ▲ 2

Congratulations and thanks a lot! @Michael
some question about "RankGauss", for feature x, "RankGuass" means
erfinv((x-Min_x)/(Max_x-Min_x)),
is it?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago ▲ 3

no. You need the rank transformation before to equalize the numeric feature regions.

liuyongqi • (120th in this Competition) • 5 years ago ▲ 1

thank you for your reply, so it is something like erfinv(rank_x/N) ?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago ▲ 1

yes exactly

Lain • (1162nd in this Competition) • 5 years ago ▲ 2

Amazing solution! Thank you very much for sharing your deep thoughts and love for representation learning. I am impressed by your straight-forward approach to "autonomous feature engineering".
I have a few questions regarding DAE swap noise generation. You choose a subset of features from another row to add noise to the dataset. Here,

- How do you select the row to get a subset of features? Do you choose multiple rows or just a single row?
- How do you apply swap noise to minibatch? Do you apply each different swap noise to each data point (i.e. select a different subset for each point in a minibatch)?

RDizzl3 • (329th in this Competition) • 5 years ago ▲ 2

@Michael Jahrer - this is definitely an amazing solution! I am on the same page with @Lain - this solution is too good not to do some personal research do you have an references where we can learn more about the noise generation schema you used? Congrats again on the 1st place finish!

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago ▲ 1

spend some time in life to think about solving problems. Often a simple approach explains the underlying

anttip • (135th in this Competition) • 5 years ago ▲ 2

Congrats. You deserved this win. Was there a reason to use vanilla SGD over ADAM?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago ▲ 3

thanks, good question. I would like to have ADAM as a switch to try it out, its not there :) I was too lazy to code it and I am only 99% sure how to implement it, I used it for another sgd model in another domain .. another story. Anyway. ADAM needs additional memory storage of the weights, you need to store 2 copies of the weights for all this moment estimates. A problem for really large nets, but doable. Being too happy with vanilla SGD, works just fine.

anttip • (135th in this Competition) • 5 years ago ▲ 0

It's definitely worth implementing. I notice you have very large epoch numbers (300, 1000). With decaying SGD this is necessary, but with adaptive SGDs such as ADAM you could do with a fraction of the epochs. NADAM works slightly better than ADAM, but it hasn't become popular.

Also, simple step decay SGD will not reach even the local optima, if some parameters have very different scales than others. Probably that was not the case with these models.

YPS9 • (1029th in this Competition) • 5 years ago ▲ 2

The way you have organized your approach and solution explains your 1st position, Great Solution !!! Congrats !!!

James Trotman • (80th in this Competition) • 5 years ago

2

> **Michael Jahrer wrote**
>
> ... Models run longer than I spend on writing code.

This is hopefully the aim and ultimate reward for every Kaggler - finding an optimization plane where you improve your results whilst minimizing the time you have to spend.

I love the inputSwapNoise idea, a very elegant way to reduce the noise level to one parameter.

Congratulations on a job well done. This is a standout result that will really stand out a long time.

(P.S. Did your linear blend really take 36 seconds? One possible area for improvement :-P)

Roberto Spadim • (2242nd in this Competition) • 5 years ago

2

wow! i will get a beer and a coffee to read this! thanks and congrats!!!!

Zafar • 5 years ago

0

I think XGBoost is worth a try!

Iuri Ferreira • 5 years ago

0

Hi

Ennn • (1243rd in this Competition) • 5 years ago

0

Congrats! and thanks for the explanation

ValeriyBabushkin • (397th in this Competition) • 5 years ago

0

Denoising autoencoders (DAE) are nice to find a better representation of the numeric data for later neural net supervised learning.
Thanks for info
As far as I know - it makes sense to take a represantation from the most narrow layer in bottleneck AE as a new set of features instead of original one
What layer output has been used as a new set of features from DAE? Was it the same layer in any one of them

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

8

yes basically you can use any layer activations as new dataset. Information is present is all layers. When using all layers you don't loose any info (this was my idea, I use ALL layer outputs as features) but maybe not needed. But haven't tried only taking eg. last hidden layer in a deep stack setting.

ValeriyBabushkin • (397th in this Competition) • 5 years ago

-1

Cool - thank you
You mentioned you have used linear activation function - e.g. - just input and weight multiplication
Did you use it for all (any specific reason?) or only for bottleneck layer?

And this layer activations - did you just concat them in one huge dataset as you mentioned of 1-10k dimension?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

13

I recommend linear activation in the middle layer of bottleneck setup because relu truncate the values <0. Yes just concat to a long feature vector. Here for a deep stack DAE 221-1500-1500-1500-221 you get new dataset with 4500 features.

Zeeshan-ul-hassan Usmani • (19th in this Competition) • 5 years ago

0

Thanks for Sharing. That's the spirit of Kaggle. Brilliant work, we tried our best to get into your mind and find out what you did. :-)

GOGGLES • a month ago

Thanks for sharing !

RuiYang Ju • a month ago

Good sharing and thanks a lot, bro

Saad Azam • 3 months ago

I have read this like 3 times and I might read this a couple more times. I am amazed by the ideas you brought and implemented. Thank you so much for sharing this.

JhonesDuran • 4 months ago

Very helpful! Thanks for sharing

ThanhThong • 4 months ago

Thank you for sharing

Campion • 6 months ago

This is really helpful!! Thanks for sharing!👍👍

Chocolateman • 7 months ago

Great, Thank you!

leeeein • 7 months ago

I dislike complex feature engineering either, but I often perform worse with representation learning than feature engineering. Is there some criterion for determining whether a dataset is suitable for learning-based feature engineering?

Danila Berezin • 8 months ago

Great!! Thanks

Mihai Preguza • 9 months ago

Thank you for sharing!

Balu Nair • 9 months ago

Thanks for sharing

zez • 9 months ago

Thanks for sharing

Vitali Avagyan • 10 months ago

Thanks for sharing your ideas, very useful indeed!

SladeF • a year ago

This was such an interesting and informative read, thank you for this!
My one question is, would you recommend learning how to use ML in C++/Cuda?

Mark Babayev • a year ago

I think the "RankGauss" can be created with a built-in sklearn transformer:
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html

```
qt = QuantileTransformer(output_distribution="normal")
```

Please correct me if I'm mistaking.

DavidCastillo • 2 years ago

Thanks for sharing, recently I've started to try with DAEs and your advices will help a lot. 👍👍

Sujit • 2 years ago

0

Thanks for sharing very helpful

Nhị Thanh Trần Hồ • 2 years ago

0

Great work!!!

Mahmut Sami EROGLU • 2 years ago

0

Thanks for sharing your knowledge!!

Suraj Kumar • 2 years ago

0

Thanks for sharing!

turbcool • 2 years ago

0

Nice work!

denbite • 2 years ago

0

Nice work, thx

ShanglinLi • 2 years ago

0

I've learnt a lot, thank you!

Issei Kobayashi • 2 years ago

0

Thanks for sharing your great knowledge!!

Andrey Kuzmenko • 2 years ago

0

Great post ! Thanks for sharing.

Kavyashree B.S • 2 years ago

0

Thanks for the information

Nags Data • 2 years ago

0

Great! thanks for the sharing the knowledge.

Moore • 2 years ago

0

Thanks for sharing. Very helpful.

varad • 2 years ago

0

thanks for sharing

Dipin P Joseph • 2 years ago

0

Thanks for Sharing!

Alex Mircea • 2 years ago

0

Impressive!

Temur Utamuradov • 2 years ago

0

Interesting

PRANAV TOMAR • 2 years ago

👌👌 Perfect.

0

Souham Ghosh • 2 years ago

Thank you for the insights.

0

Jai Joshi • 3 years ago

wow thanks

0

Karthick • 3 years ago

thank you

0

Chai Shuwen • 3 years ago

Thanks! Great sharing

0

ironorih • 3 years ago

great read

0

Ash • 3 years ago

Interesting!

0

Mauro Llanos • 3 years ago

Interesting

0

Gaurav Dudeja • 3 years ago

Thanks

0

Maaheen Jaiswal • 3 years ago

Nice

0

Kuldeep Arya • 3 years ago

great

0

Guanwen Wang • 3 years ago

Great Job! Thank u!

0

lemonlemon4 • 3 years ago

Amazing work! thanks!

0

SangKyu • 3 years ago

Thank you!

0

Jitendra Singh Malik • 3 years ago

nice

0

Jitendra Singh Malik • 3 years ago

interesting

0

Max Konyaev • 3 years ago

Nice!

cccache • 3 years ago

Good job. Learned a lot.

RAJESH_PARTHASARATHY_NAMAGIRI • 3 years ago

thanks

stamacake • 3 years ago

thanks

Dima Shmundiak • 3 years ago

Good Job!

SeungWon Kim • 3 years ago

Great Post!! Thanks for Sharing :)

witlof • 3 years ago

Great post! Thanks for sharing

Ensar Erdogan • 3 years ago

Thank you very much!

songyuquan • (3558th in this Competition) • 4 years ago

Great job! I was wondering if somebody use rgf model.

JayantSachdev • 4 years ago

Thanks for Sharing

dhaqui the kaggler • 4 years ago

Thanks for sharing!

Gaurav Kshirsagar • 4 years ago

Nice work !!!

corner200 • 4 years ago

How much does this procedure improve the score?
Did someone try it in other competitions. If you use the neural network without the autoencoder input what score do you get. And what score after using the autoencoder input?

daishu • (240th in this Competition) • 4 years ago

HI, Michael. I note that you initialized weight to random_uniform[-sqrt(1.0/input_N),sqrt(1.0/input_N)], right?

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

right

daishu • (240th in this Competition) • 4 years ago

HI, Michael. Which one did you use to compute gradients, L(w+Δw)/Δw or dL/dw?
Thanks in advance.

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

0

simple backprop (vectorized over batch). So I guess its dL/dw

daishu • (240th in this Competition) • 4 years ago

0

May you kindly post your rank_gauss function?

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

1

I added reference code to transform one vector

daishu • (240th in this Competition) • 4 years ago

0

Thanks for your sharing!

imant • 4 years ago

0

Thinking about the denoising autoencoder I had a curious thought: if the data that we were given in this competition was transformed in a similar way as in the DEA (swap noise), then this might explain the effectiveness of this approach.

More concretely, the DEA could generate new samples from the same distribution, thus having more training examples to learn from. Even if that does not suffice to learn the inverse mapping (which is random), it might still increase the robustness of the model with respect to noise.

Why should we be provided with noisy data? It's just a thought experiment and I'm not accusing anyone, but one could imagine that artificial noise increases the level of anonymization and that the best results would still be useful as they represent a pessimistic bound for the true performance, which should only be better on the "original" data.

Shuo Yang • 4 years ago

0

Thanks, i have learnt a lot

Skyfacon • (1424th in this Competition) • 4 years ago

0

I am wondering how you deal with the High-dimensional category data, say if you have a categorical feature with about hundreds of classes, you one-hot encoding all of them and feed it into nn ? What's the input of your nn model? I think the input should be the hidden layers of the DAE, right? Did you put all data into DAE of just numeric data?

Roberto Spadim • (2242nd in this Competition) • 4 years ago

0

autoencoder doubt... how you select complexity? Any standard idea? What about mse threshold?
Thanks!! Congrats

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

4

2 or 3 hidden layers with 10x input size.

Roberto Spadim • (2242nd in this Competition) • 4 years ago

0

It's a "default" configuration, or something with a paper and research to explain? Or your experience gives this as "ok" to go?

Roberto Spadim • (2242nd in this Competition) • 4 years ago

1

other doubt...

the last one was about dae structure, i'm reading autoencoders (some good example: http://videolectures.net/icml08_vincent_ecrf/ but not to read) again and 2-3 layers seens to be a 'standard' approach, the size o layers too (10x) when you don't include a bottleneck at autoencoder, maybe i'm answering my question, a answer from you should be very nice to check if we converge

now about swapnoise, i'm figuring out how could be implemented, about function call

function swapnoise(input_dataset, percentage_of_random_columns, percentage_of_random_rows) output new dataset

my doubts:

1) what's the output_dataset size? length(input_dataset) * (1+percentage_of_random_rows), or you consider another input parameter? like:

function swapnoise(input_dataset, percentage_of_random_columns, percentage_of_random_rows, number_of_new_rows) output new dataset

2) when doing column swap, you first select columns to swap, or you always select new columns for each row? for example:

2.1) select random rows (considering percentage of random rows)

2.2) select random columns (considering percentage of random columns)

2.3) copy selected rows, swap selected columns with columns from any row of full dataset

or you do

2.4) select random rows (considering percentage of random rows)

2.5) copy selected rows, and *per row* select percentage of columns to swap, swap these columns with columns of any row of full dataset

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago ▲ 4

about swap noise: Neural net training happens in batches, this means the net only see a very small subset of total randomized rows, I use nearly always batchsize=128 (good compromise of speed and accuracy). Eg. with 1M rows the net processes 10^6/128=7813 batches per epoch. In an autoencoder setup input features and targets are the same. If you train with input=targets and huge hidden layers error will become nearly zero quickly (overcomplete representation, no regularizer). Swap noise on the input will modify the values in the input batch (randomly sampled from whole dataset column-wise), this acts as a very strong regularizer because the net need to "de-noise" the samples.

Roberto Spadim • (2242nd in this Competition) • 4 years ago ▲ 0

nice, i think we are converging...

i understood that you change the input variables but don't change the output (that's the idea of denoise, nice =], better than dropout that set values to 0, at least you are doing a sample in input data and not "random value" "0"), but to understand the software loops when you execute the train part (SGD):

1) epoch loop (from 0 to how many epochs you want, or maybe minimize a MSE?)

2) batch loop (from 0 to last batch, this is the default i think you don't random sample batchs, right?)

if these two loop aren't wrong...

do you execute the random sampling of columns at (2), or at (1), or maybe only once? and when you do this, you do on some columns only at the first loop, or for each loop you sample new columns? for example, at (1) you sample X columns [0,1,2,3,4,5 from 40 columns], at next epoch you sample again new columns [1,5,2,6,3,15 from 40 columns], or you still using [0,1,2,3,4] from first sample?

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago ▲ 5

1) I only use pure SGD, my experience is that you need a huge number of epoch to get good DAE representations. I recommend 1000 epochs.

2) Yes in one epoch you iterate over all data batches (see all rows exactly once). In my implementation I randomize the dataset before and then train always on the same batches. Good.

For swap noise on the inputs (targets are not changed, right!) I always sample new rows in every batch from the same column, this gives most randomization. Always generate new random numbers, I think this is important to fully explore the noise. So when u process a batch select eg. random 15% of the columns and exchange them with a value from a random row.

Roberto Spadim • (2242nd in this Competition) • 4 years ago ▲ 0

Nice! Yeap noiae is very important that how you "augment" the tabular data

Nice to know that the random is always in the batch level, output is always fixed, just input is noised

Well i think we converged :) thanks a lot i will try to implement this and check what happen, batch size noise is something not common on keras or python packages, at least to me, I will try to understand keras and if possible post a similar inplementation just to everybody understand

Roberto Spadim • (2242nd in this Competition) • 4 years ago ▲ 3

well if i understood it right =) here it is:

```
# work did by Roberto Spadim, talking with Michael Jahrer at kaggle Porto Seguro Competition
# congrats Michael!
```

https://www.kaggle.com/rspadim/simple-denoise-autoencoder-with-keras

Roberto Spadim • (2242nd in this Competition) • 4 years ago ▲ 0

michael, about python expert. just learn numpy, matplotlib, keras, lightgbm, xgboost, tensorflow, mxnet, scipy is nice too. that's all i use. ide you can use pycharm or spider, it have a nice integrated debugger blablabla =)

use. Ide you can use pycharm or spider, it have a nice integrated debugger blablabla =)

thanks =) you are wellcome, congrats again

Roberto Spadim • (2242nd in this Competition) • 4 years ago 0

@mjahrer, one more doubt...

when you start DAE, what kind of weights you use to start? random weights? lecunn? anyother?

i'm ending your solution using keras here and i checked that with small dataset DAE diverge (probably RELU got crazy) :

https://www.kaggle.com/rspadim/1st-pos-trying-to-reproduce-with-keras

trying to write your solution on keras + python was a nice task :)

nakagawa • 4 years ago 0

Thanks for sharing your solution.

Jack • 5 years ago 0

Congatulations and thanks for sharing your solution !

Lo Chenchou • 5 years ago 0

Amazing!
Thanks for sharing

Atul • 5 years ago 0

Thanks for sharing !!!

veerendra • 5 years ago 0

Congrats and thanks for sharing.

yliu • 5 years ago 0

mark

Roberto Moura • 5 years ago 0

Congrats Michael

KaizaburoChubachi • (1393rd in this Competition) • 5 years ago 0

Hi everyone,

I am trying to reproduce the brilliant works.
Though I implemanted #2 model using python, it does not work well.
I will share my code on the kernel, and I am very happy if you point out my mistake :)

https://www.kaggle.com/zaburo/wip-reproduce-1st-place-solution

Thanks.

riki • (754th in this Competition) • 5 years ago 0

@michael jahrer: would be great if you can let us know that for the DAE, what MSE value were you able to reach in the 1000 epochs?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago 3

ok, I attached a file in my original post, its the console log of model nr 3. If you do a "cat nn.cfg.log" from a Linux console you get the output with nice colors. Maybe somebody can read something out of it. You know the MSE depends largely on the processed input data.

riki • (754th in this Competition) • 5 years ago 1

Yes, trying to follow the same process that you described in your writeup. Thanks for the file.

riki • (754th in this Competition) • 5 years ago

1

if the inputs to the NN are hidden layer activations of the DAE, then all the inputs to the NN are >=0 since you are using ReLU in the DAE hidden layers. is this correct or did i miss something?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

1

correct.

Don • 5 years ago

0

Michael - wondering if you would be interested in helping me to validate using AutoML on this dataset? We've built an AutoML framework and would like to give it a shot on others. You are welcome to use our GPU's for free for testing :) . Contact me if interested in working with me.

Here's a public access to the example Jupyter notebook:

https://c.onepanel.io/onepanel-demo/projects/automl/code

mtax • 5 years ago

0

Hi Thanks for sharing!

Lifan Zhang • 5 years ago

0

Thanks for sharing!

ireallyloveyou • 5 years ago

0

For model 4，what is the colGroups meaning?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

0

with colGroups I tried to swap columns with same meanings. This concerns categoric cols. I tried to swap the interger category+corresponding one-hot cols at once. No success.
I answered a similar question in one of the comments above..

onlyzs • (1625th in this Competition) • 5 years ago

0

great job!

Julio Meza • 5 years ago

0

Nice

Weiss • 5 years ago

0

Thanks for sharing!

George Bogodukhov • 5 years ago

0

Very exciting, thanks for sharing!

Mohamed Nassef • 5 years ago

0

Impressive work .. my first read on kaggle .. well done !:)

Mohnish Raj • 5 years ago

0

Great work ! congratulations !

ImmanuelWilliams • 5 years ago

0

What made knn not work?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

0

it dont blend, at least my implementation does not contribute to the ensemble.

[Deleted User] • 5 years ago

0

Great! Thanks for sharing!

Sundar Krishnan • (3666th in this Competition) • 5 years ago

0

Congrats Michael.

Minjun Wang • 5 years ago

0

Great job done!

daishu • (240th in this Competition) • 5 years ago

0

Thanks! May I repost to my blog?

Nicolas Marticorena Vidal • 5 years ago

0

thx for sharing

Jason Joslin • (3554th in this Competition) • 5 years ago

0

Awesome Michael. Thanks for sharing.

YaGana Sheriff-Hussaini • (62nd in this Competition) • 5 years ago

0

Thanks so much for sharing your solution and thoughts in a detailed and really useful fashion. I share you point about feature engineering as I am in the group whose creativity is on the slow side for an average competition life span. My best efforts so far in my short Kaggling life was in the Instacart competition and I was no where near finished by the time it ended. Plus I believe we can automate feature engineering with NN and other clever techniques.

Using Denoising autoencoders (DAE) to get features for supervised learning is really a clever idea I have not thought about. I also like the fact that you tried GANs with this data even if the results were not comparable to your other models. All this shows you deserved to win this competition.

Congratulations and thank you.

Alex Shieu • 5 years ago

0

Thank you for sharing!

Seongwon Jang • 5 years ago

0

Thank you for sharing. Could you tell me which activation function have you used in deepstack AE and bottleneck AE? You mentioned you have used linear activation function in the middle layer of bottleneck. Did you use it for all (any specific reason?) or only for bottleneck layer? If you have used linear activation function only for bottleneck layer, which activation function have you used at other layers? I look forward to your reply. Thanks.

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

3

yes linear within bottleneck layer. Other layers relu.

Seongwon Jang • 5 years ago

0

But, In deepstack AE, if you use relu function between last hidden layer and output layer, output activation nodes have only positive value(x>0). But I understood that input values after being normalized by rankgauss have negative values.

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

3

yep. Output layer is always linear in my DAE experiments.

jason • (359th in this Competition) • 5 years ago

0

thanks for sharing your wisdom

Yaoxiang Li • 5 years ago

0

Congratulations! Michael.

Future • 5 years ago

Thank you for sharing your approach!

Tinku • (3331st in this Competition) • 5 years ago

Thanks for sharing the detailed approach.. Michael.. I am not starting with NNs anywhere soon. But felt happy on starting to get interested.

YouHan Lee • (2962nd in this Competition) • 5 years ago

Thanks for sharing your amazing work!

TensorFrozen • (212th in this Competition) • 5 years ago

How much did you gain by removing "calc" features ?

Nassim Zaagoub • (448th in this Competition) • 5 years ago

Congratulations, well done. Thanks for sharing the details of your approach.

Kerem Turgutlu • (1088th in this Competition) • 5 years ago

Hi Michael, did you corrupt variables at each batch on the fly with shuffle or before training stage for only once ?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

at each batch. Each batch samples new noise from the complete dataset. Complete dataset is train+test features. Copy the features before to use as target. Targets should be the clean uncorrupted features.

Dan Ofer • (2033rd in this Competition) • 5 years ago

PS: the noise trick sounds familiarly like marginalized denoising autoencoders (vanilla) / https://github.com/phdowling/mSDA

Dan Ofer • (2033rd in this Competition) • 5 years ago

This is extremely interesting!
That said, i'm somewhat astounded that the single best model was just a vanilla GBM (well, lightGBM) with some manual features dropped and some tuning. (Especialyl given how everyone and their aunt was building such models + NNs).
Then again, the differences on the LB are quite small in absolute terms, so maybe I'm just not appreciating the benefits of relative ranking :)

Any chance for python code by some eager fast.ai/kaggler/keras fan? This seems relatively trivial to add per column.

Bongo • (2155th in this Competition) • 5 years ago

Congrats and thanks for your consistent contribution to the Kaggle community! Do you have any good resources on learning DAE? where did you learn how to construct the DAE?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

My interest in all kind of nn things last back 15 years, I tried everything and still scratching only the surface. A good paper is:
Scheduled denoising autoencoders

Rick Wu • (319th in this Competition) • 5 years ago

Very interesting! Thank you for sharing.

Kerem Turgutlu • (1088th in this Competition) • 5 years ago

So I've been reading about DAEs and I have a confusion about denoising part. It's very clear that we use stochastic swapinput for X (input matrix), but don't we also need denoising for each hidden layer ? And if we do, is it going to be with the same technique? Thanks so much !

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

Just noise at input is enough. The noisy samples are look different for every batch, and the DAE learns from millions of batches. Its nearly always a different noisy version of input features that the DAE tries to reconstruct. It learns an internal representation of the prob distribution of the dataset. The internal state is used later to learn supervised from.

Yes you can add noise to hidden layers too, havent tried this.

Ofd • (1221st in this Competition) • 5 years ago

0

Congratulation Michael, a question please, how did you deal with imbalance? thanks

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

2

basically I do nothing. If we have 3.6% ones, just use them as it is. We cannot generate more samples, not more info available, thats all we have. Yes I tried upsampling in supervised nn as other kernels suggested, but had no success that improved the blend, I guess I could improve individual models with upsampling but only a little.

seapea • 5 years ago

0

I like this write-up. Do you think this kind of hardware is necessary? I mean, I have a good machine, but I still wonder.

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

0

32GB RAM is minimum for 5CV on 4500 features with this data size. Of course you can use an older GPU, will run slower.

ZifengWang • (2036th in this Competition) • 5 years ago

0

brilliant!

Emerson Bertolo • (1386th in this Competition) • 5 years ago

0

Thanks Michael for sharing it. It was an impressive result with a so easy to understand estrategy. And doing it from scratch with C++ is really inspiring!

Hossein Amirkhani • (266th in this Competition) • 5 years ago

0

Congratulation! It was impressive and insightful. Is it possible to release the submission files for each of your six models?

Sidharth • 5 years ago

0

very nice!!!!

Lesaffrea • (1405th in this Competition) • 5 years ago

0

Thanks Michael to share. Great Insights

Serigne • (175th in this Competition) • 5 years ago

0

Congrats Michael for your amazing winning and keeping the 1st place from the beginning to the end.

And thanks for sharing this incredible solution.

Jorge Humberto • (1085th in this Competition) • 5 years ago

0

Thanks Michael for sharing your solution, certainly I going to learn more from it.

skw1990 • (2446th in this Competition) • 5 years ago

0

Congratz - may i know why you removed *calc variables?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

2

suggestion from public kernels. Improved my CV. So I removed *calc

Thomas Fischer • (2907th in this Competition) • 5 years ago

0

Congratulations. Just impressive. And beautiful.

Marcelo Senaga • (1368th in this Competition) • 5 years ago

0

Thanks for sharing this!

[Deleted User] • 5 years ago

0

A nice learning resource this is...thank you.

Niranjan Nakkala • (565th in this Competition) • 5 years ago

0

Leader knows the end results, you are confident about what you are doing from day1 to till end of competition.you proved system to automatically discover the representations needed for feature detection or classification from raw data performs better. Congratulations and Thank You Michael for sharing your Approach.Good Luck. :)

Ihara Takumi • (1816th in this Competition) • 5 years ago

0

Thanks for sharing your great job.
I do my best to be like you!!

Safayet Karim • (1006th in this Competition) • 5 years ago

0

Thanks for sharing.

Shabie Iqbal • (48th in this Competition) • 5 years ago

0

Well many people got one more prediction right: Michael Jahrer is doing something different. Well no shit!

lll • (724th in this Competition) • 5 years ago

0

Congratulations! Two quick questions,
1.

> Adding gaussian or uniform additive / multiplicative noise is not optimal since features have different scale or a discrete set of values that some noise just didnt make sense.

You have normalized features using RankGauss before putting them into DAE in the previous part, right? So in my mind they will share the same scale, please correct my misunderstanding :)

1. Does it make sense to use output of DAE and put it into other non-deep learning algorithms like XGBoost or Logistic Regression?

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago

2

hi, yes scale should be nearly equal. But still the problem of mixed numeric/1-hot columns. There's only one answer :D .. try it out. Yes makes sense, you can put in DAE features to xgboost/lightgbm. It improved my CV a little but get worse on public LB, so I rejected it (BIG mistake).

KaizaburoChubachi • (1393rd in this Competition) • 5 years ago

0

Would you tell us the scores of DAE + xgboost/lightgbm on CV, public LB, and private LB?

PrinceThomas • (1544th in this Competition) • 5 years ago

0

Congratulations Michael and thank you for sharing the winning solution. Lot of inputs for me to learn. Thanks a ton for the same.

Chia-Ta Tsai • (264th in this Competition) • 5 years ago

0

Impressive solution and great write-up. Thanks for demonstrating the amazing potentials of deep learning.

Ananth K • (263rd in this Competition) • 5 years ago

0

Michael, this is an amazing solution! thank you for sharing.

meikun • (1176th in this Competition) • 5 years ago

0

Amazing work. Thank you Michael. I have learned a lot from this competition and discussion!

QICHENHU • (369th in this Competition) • 5 years ago 0

Amazing! Thanks for sharing!

visualmind • (398th in this Competition) • 5 years ago 0

Brilliant solution!!! Congrats and thank you for sharing in such a detail. Applying unsupervised learning is truly an amazing approach. You won the competition elegantly and truly deserve the no.1!!

terenceflow • (1212th in this Competition) • 5 years ago 0

Amazing work! Thanks for sharing.

Yang • 5 years ago 0

Beautiful，congrats to the first place and thanks for the sharing.

Robin Smits • (1603rd in this Competition) • 5 years ago 0

Very impressive solution! For me as a starting Kaggler this is truly amazing to see what you did. Congrats!

Storch • (427th in this Competition) • 5 years ago 0

Thank you for sharing your experience with us. I really learned a lot on how to use NN effectively with denoising autoencoders.

Witold Oleksiewicz • (50th in this Competition) • 5 years ago 0

Congrats Michael

mlhammer2logit2quit • (1558th in this Competition) • 5 years ago 0

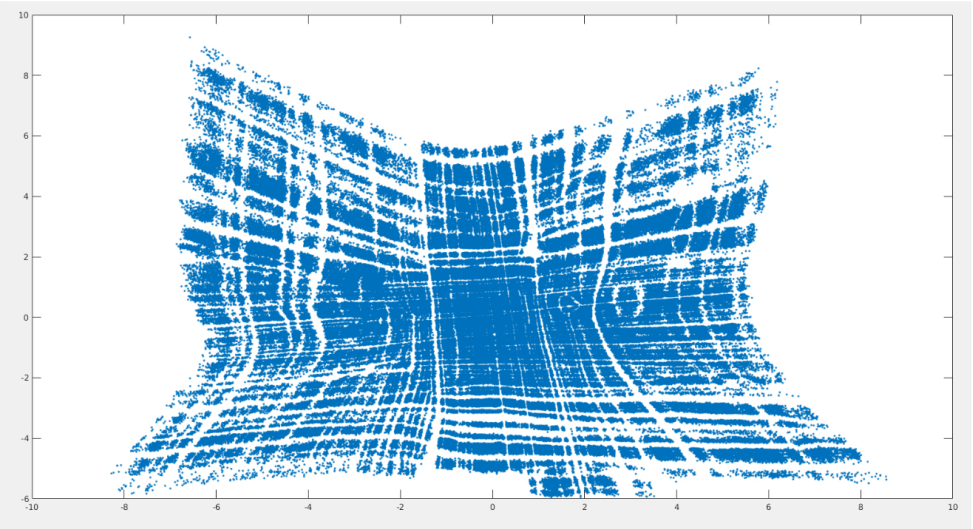
Congrats and thanks for sharing your fantastic approach.
With hindsight, "swap noise" seems natural considering the signal-to-noise some have studied:

"many people who are very similar in terms of their insurance parameters, yet some of them will file a claim and others will not"

Your regularization was the "other technology" we suspected. Many hedged by emphasizing model diversity, but apparently a more economical use of data was the better instinct for this kind of problem. A well designed test set made unsupervised learning especially effective.

Michael Jahrer Topic Author • (1st in this Competition) • 5 years ago 9

thanks, yes I noticed the nice tsne figures by Tilii, very well done! During competition I also tried to map the data to 2D space just for curiosity with a bottleneck DAE, see:



mlhammer2logit2quit • (1558th in this Competition) • 5 years ago 1

I'm curious about the performance of KNN on this projection. I had experimented with Linear Discriminant Analysis for supervised dimensionality reduction for KNN.

Joe Fox • (2679th in this Competition) • 5 years ago

0

Great work..well done.

steelrose • (21st in this Competition) • 5 years ago

0

congrats to the first place and thanks for the sharing, have a lot to learn from you

鼠(China) • (183rd in this Competition) • 5 years ago

0

simple and powerful, a lot to be learned!

Ankit • (1939th in this Competition) • 5 years ago

0

Thanks for sharing!

Andy Harless • (21st in this Competition) • 5 years ago

0

[Obligatory pun about "auto encoders"]. Also, congratulations!

Heads or Tails • (299th in this Competition) • 5 years ago

0

Great write-up; lots of knowledge and inspiration! Thanks so much and congratulations again!

virtusman • (2337th in this Competition) • 5 years ago

0

Very interesting! Congratulations on your win and thanks for sharing!

This comment has been deleted

This comment has been deleted

This comment has been deleted

This comment has been deleted

This comment has been deleted

This comment has been deleted

This comment has been deleted

This comment has been deleted

Daniel Möller • (3595th in this Competition) • 4 years ago

0

I think the best for these NN tasks is to use high level libraries such as Keras or PyTorch.

Roberto Spadim • (2242nd in this Competition) • 4 years ago

0

python is single thread, you have a bottleneck at python side, i prefer a c/c++ to high power cuda, but python is ok to test

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

2

2013 I bought a book "CUDA Programming: A Developer's Guide to Parallel Computing with GPUs (Applications of Gpu Computing)"

for intro. But major progress was made with self-experiments. I run kernels on GPU and exactly compare the CPU version, where I know its calculated correctly. Big difficulty was to understand the grid layout on a GPU (blockIdx, threadIdx..) to efficiently parallelize. Made many experiments on runtime and howto grid the problem (eg. calculate dropout or activation functions). All done in C++ with cuda-toolkit and gcc.

YaGana Sheriff-Hussaini • (62nd in this Competition) • 4 years ago ^ 0

Thanks @mjahrer, this book sound interesting, I will look for it.

This comment has been deleted

This comment has been deleted

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago ^ 8

we all know neural nets need a proper normalization on numeric cols to learn coz they calc gradients directly from inputs. In books you find mean/std or min/max normalization, which are not optimal when values are not gaussian distributed. During many years I tried many things, for me this "RankGauss" works well out-of-the-box for every dataset. Recently I found out that using the target information too leads to a better rank transformation. The idea here is to give the input range more weight when target changes often (more interesting regions). You see there is still room for creative approaches in normalizing data for neural nets.

This comment has been deleted

Oscar Takeshita • (69th in this Competition) • 4 years ago ^ 0

@Micheal Jahrer. First thank you for continuing providing insights even after the competition has already long been completed. Do you recommend any papers on the subject? Does it have anything to do with maximizing entropy?

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago ^ 7

no, I'm not aware of any papers about normalization. But I am away from academics since many years. I do not have a good overview of recent work in this field. My feeling is that nn cannot split in arbitrary fine granularity like gbdt(xgboost) does therefore you need to guess "good input regions"

This comment has been deleted

Oscar Takeshita • (69th in this Competition) • 4 years ago ^ 1

Thank you. Then maybe we could translate the splitting points or regions found by a GB tree method and use it as an input pre-processor to the nn.

This comment has been deleted

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago ^ 1

try it on your own on a particular dataset, CV will give you a hint which might work better

This comment has been deleted

This comment has been deleted

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago ^ 1

I think I need to attach reference c++ code for this transformation

Roberto Spadim • (2242nd in this Competition) • 4 years ago ^ 0

If i understood it right, the idea is change the distribuition of these variables, doing a rank transformation and after feeding it to nn or executing a post transformtion, something like: variable distribution → uniform distribution→ normal distribution (just an example)

Oscar Takeshita • (69th in this Competition) • 4 years ago ^ 0

What about using Tom's python implementation in this thread?

This comment has been deleted

This comment has been deleted

Shabie Iqbal • (48th in this Competition) • 4 years ago

0

@Oscar By using the GB-tree, you mean segmented "normalization"? I think that defeats the very pupose of normalization... No?

Shabie Iqbal • (48th in this Competition) • 4 years ago

1

@Michael Jahrer "we all know neural nets need a proper normalization on numeric cols".

Does this mean you **ALWAYS** normalize inputs for NNs i.e. even word vectors, DAE representations etc.?

Michael Jahrer Topic Author • (1st in this Competition) • 4 years ago

1

very often normalization it does not hurt. A few exceptions: 1-hot category columns are ok to put in as 0 and 1. Outputs from DAE (activations from hidden units) surprisingly doesn't need normalization. Image pixels (uchar) work well by divide them by 255.

Shabie Iqbal • (48th in this Competition) • 4 years ago

0

So to summarize (my very limited understanding):

1. Since word vectors are effectively encoders then those too I guess don't particularly need normalization. It has worked for me at least. I guess perhaps encoder outputs are relative (i.e. share a common space in an X-dimensional space) and hence the deviations in one or the other dimension is information for the NN.
2. As for tabular numeric data, sure it needs normalization. I guess essentially, DAE in your solution is to numeric data what word vectors are to text barring the obvious dissimilarities between text and numbers. DAEs don't need normalization.
3. OHE doesn't need normalizing but pixel values do.

Thank you very much for such honest and detailed feedback. I learnt a lot really. Hope to do more. Now I am wondering what's the secret sauce of your model because... given the computing power definitely looks replicable given the generous details ;)

Edit: God I used a lot of "I guess"

Oscar Takeshita • (69th in this Competition) • 4 years ago

1

@ S.Iqbal After reading Michael's comment, I was just wondering if there might be normalization methods that could be guided by using GB-tree.

> **Michael Jahrer wrote**
>
> My feeling is that nn cannot split in arbitrary fine granularity like gbdt(xgboost) does therefore you need to guess "good input regions"