Report. Valeev Aidar.

User manual

Code can be found here.
Install all dependencies (tensorflow==1.15)
Run all cells in the jupyter notebooks.

Task 1. Iris dataset

Feed Forward NN

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 17)                85
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 126
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 24
=================================================================
Total params: 235
Trainable params: 235
Non-trainable params: 0
```

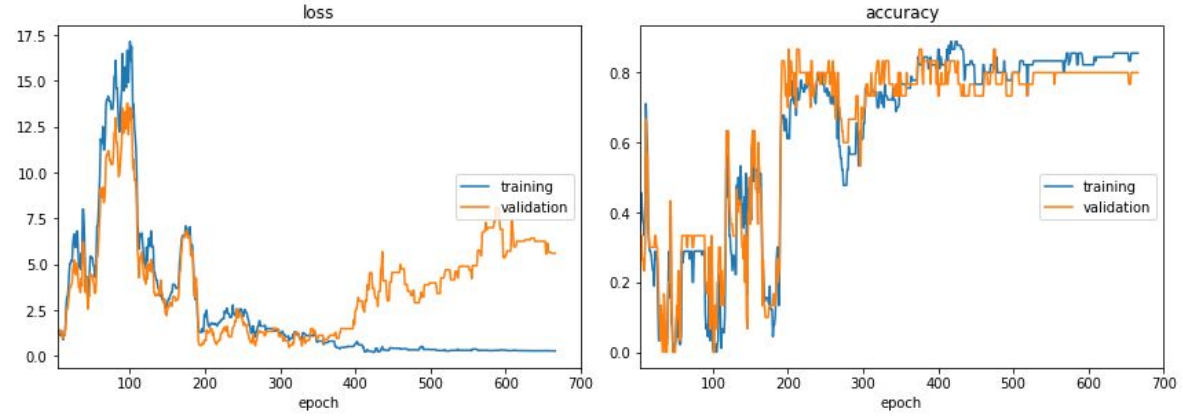
```
def create_model():
    model = Sequential()
    model.add(Dense(17, input_dim=input_dim, activation='relu'))
    model.add(Dense(7, activation='relu'))
    model.add(Dense(output_dim, activation='softmax'))
    return model
```

```
model.compile(loss='categorical_crossentropy', metrics=["accuracy"])
```

Results:

Parameters	T=5, iterations=700, a=0.99				
Trials	1	2	3	4	5
Loss	0.5390	0.3024	1.8085	0.1348	2.4077
Accuracy	86.7%	83.3%	80.0%	90.0%	83.3%
Time	7.06s	8.87s	8.55s	9.5s	9 s

```
model = simulated_annealing(T=5, iterations=700, a=0.99)
```



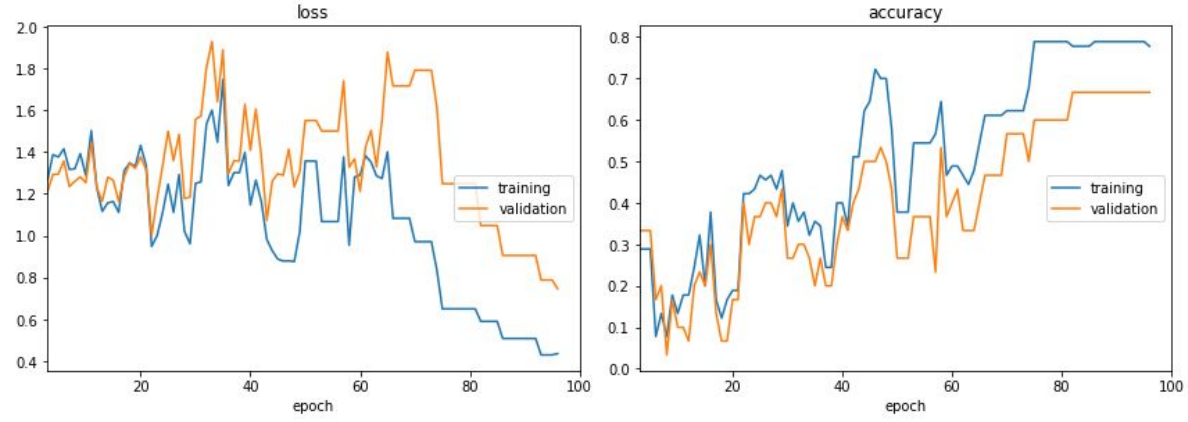

```
loss:
training   (min:    0.207, max:   17.131, cur:    0.272)
validation (min:    0.464, max:   13.765, cur:    5.594)

accuracy:
training   (min:    0.000, max:    0.889, cur:    0.856)
validation (min:    0.000, max:    0.867, cur:    0.800)

Test loss: 2.1830
Test accuracy: 83.3%
```

Parameters	T=100, iterations=100, a=0.9				
Trials	1	2	3	4	5
Loss	0.5017	2.0874	0.3832	0.5254	1.4822
Accuracy	83.3%	36.7%	86.7%	80.0%	70.0%
Time	1.35s	2.18s	2.21s	2.29s	2.45s

```
model = simulated_annealing(T=100, iterations=100, a=0.9)
```



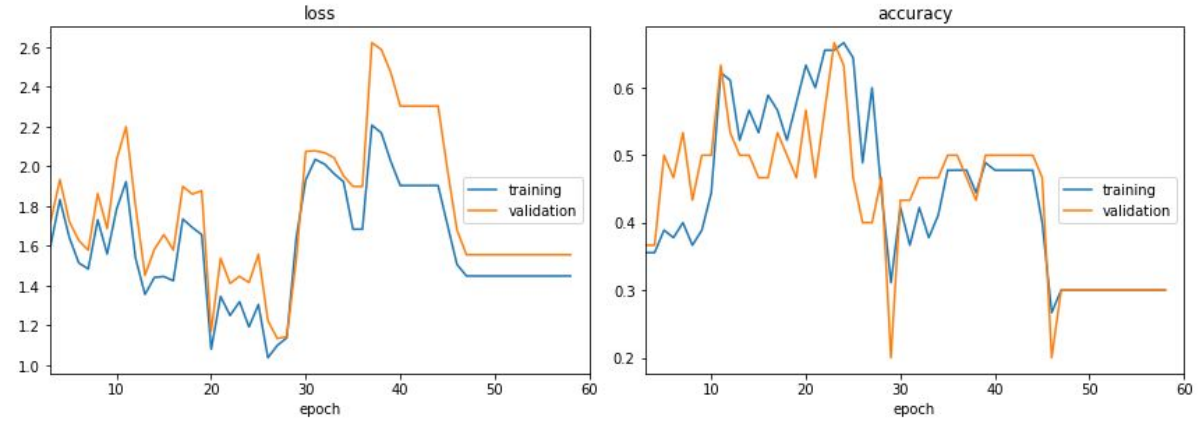

```
loss:
training   (min:    0.430, max:    1.747, cur:    0.437)
validation (min:    0.747, max:    1.928, cur:    0.747)

accuracy:
training   (min:    0.078, max:    0.789, cur:    0.778)
validation (min:    0.033, max:    0.667, cur:    0.667)

Test loss: 0.8550
Test accuracy: 56.7%
```

Parameters	T=1e8, iterations=60, a=0.6				
Trials	1	2	3	4	5
Loss	3.5288	2.4013	4.0487	2.4537	6.6126
Accuracy	23.3%	20.0%	30.0%	23.3%	26.7%
Time	1.1s	1.88s	1.99s	2.05s	2.08s

```
model = simulated_annealing(T=1e8, iterations=60, a=0.6)
```



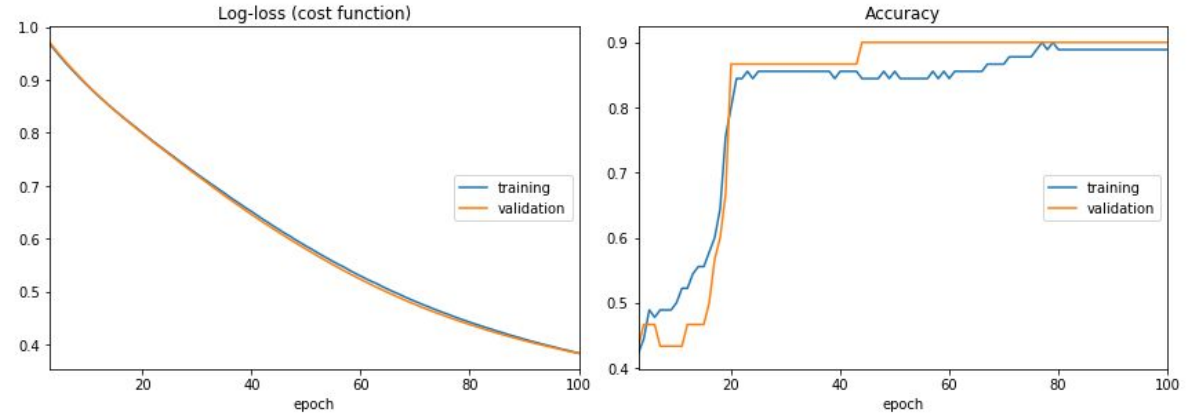

```
loss:
training   (min:    1.038, max:    2.207, cur:    1.449)
validation (min:    1.134, max:    2.622, cur:    1.556)

accuracy:
training   (min:    0.267, max:    0.667, cur:    0.300)
validation (min:    0.200, max:    0.667, cur:    0.300)

Test loss: 1.3687
Test accuracy: 30.0%
```

Parameters	SGD				
Trials	1	2	3	4	5
Loss	0.4013	0.3528	0.3112	0.4440	0.3676
Accuracy	83.3%	93.3%	93.3%	83.3%	83.3%
Time	1.03s	1.58s	1.61s	1.72s	1.66s

```
model = fit_evaluate_optimizer('sgd')
```



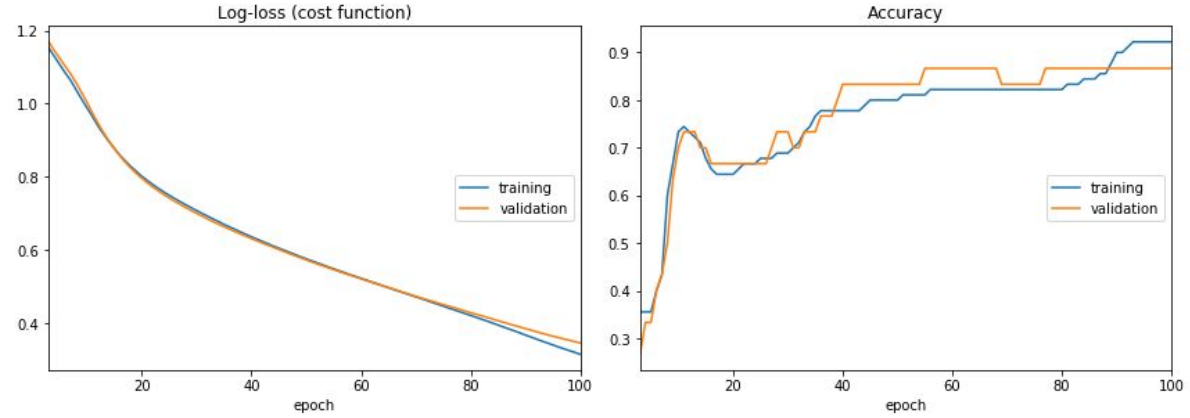

```
Log-loss (cost function):
training   (min:    0.384, max:    0.996, cur:    0.384)
validation (min:    0.383, max:    0.998, cur:    0.383)

Accuracy:
training   (min:    0.378, max:    0.900, cur:    0.889)
validation (min:    0.400, max:    0.900, cur:    0.900)

Test loss: 0.3446, Test accuracy: 90.0%
```

Parameters	Adam				
Trials	1	2	3	4	5
Loss	0.2297	0.2811	0.2019	0.1928	0.2816
Accuracy	90.0%	86.7%	96.7%	93.3%	90.0%
Time	1.5s	1.86s	1.92s	2.08s	2.18s

```
model = fit_evaluate_optimizer('adam')
```




```
Log-loss (cost function):
training   (min:    0.314, max:    1.201, cur:    0.314)
validation (min:    0.345, max:    1.217, cur:    0.345)

Accuracy:
training   (min:    0.356, max:    0.922, cur:    0.922)
validation (min:    0.267, max:    0.867, cur:    0.867)

Test loss: 0.3142, Test accuracy: 90.0%
```

Comparison

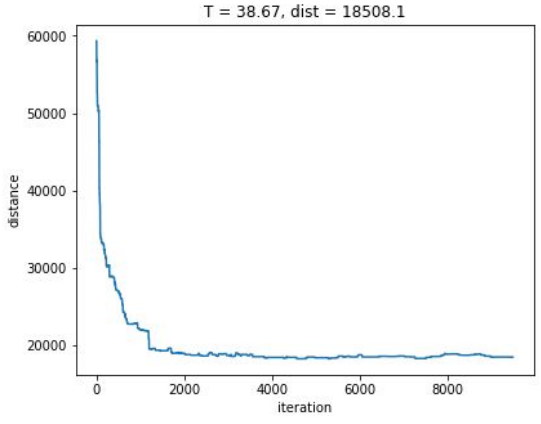
Average results:

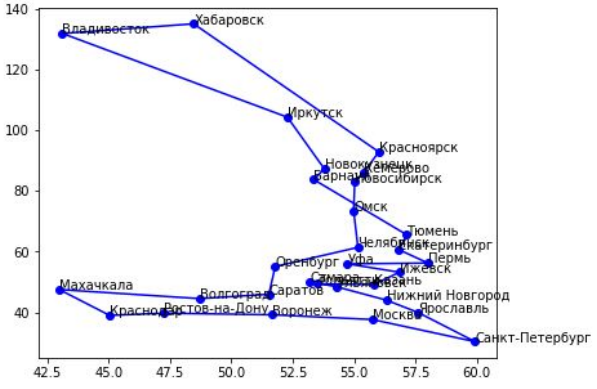
	SA T=5 iters=700 a=0.99	SA T=100 iters=100 a=0.9	SA T=1e8 iters=60 a=0.6	SGD	Adam
Accuracy	84.7%	71.3%	24.7%	87.3%	91.3%
Time	8.6s	2.1s	1.82s	1.52s	1.9s

The accuracy of the simulated annealing optimized models decreases dramatically as alpha, which is annealing rate, decreases. The best accuracy among them belong to the leftmost model, which falls behind both SGD and Adam, however taking 4-5 times more time to compute.

Task 2. Combinatorial optimization

Parameters	T=100, iterations=10000, a=0.9999				
Trials	1	2	3	4	5
Distance	18983.3	19156.7	18583.6	19115.6	20157.4
Time	405ms	296ms	290ms	293ms	265ms

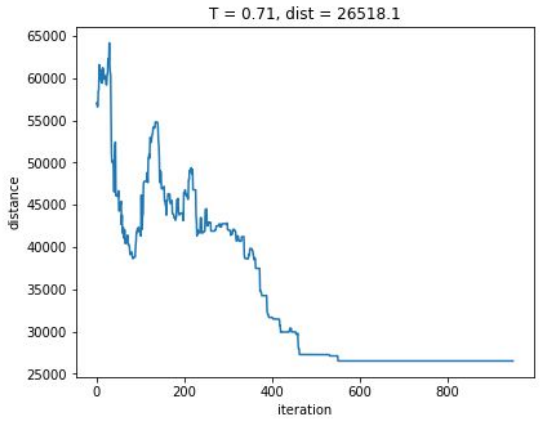


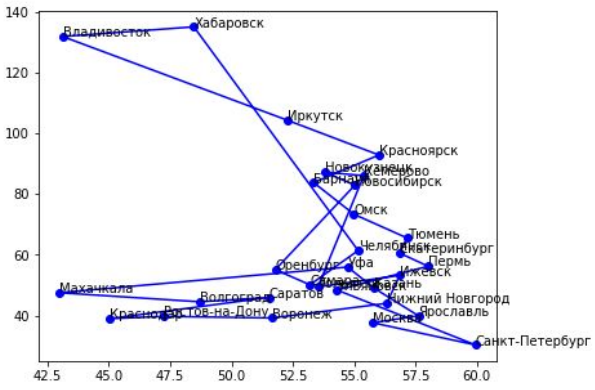



```
Final distance: 18503.8
```

Parameters	T=1e4, iterations=1000, a=0.99				
Trials	1	2	3	4	5

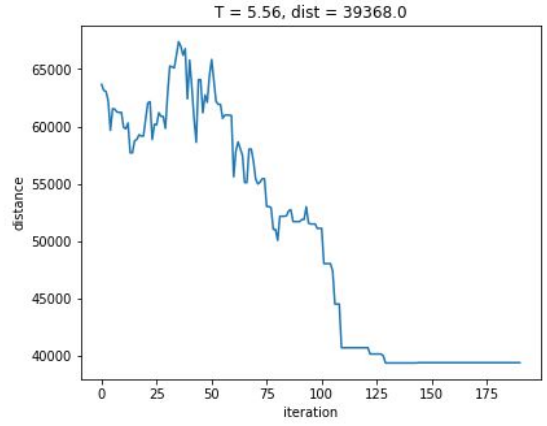
Distance	29352.4	25452.2	25061.9	26250.3	26059.7
Time	41.3ms	39.1ms	40.3ms	40.6ms	51.4ms

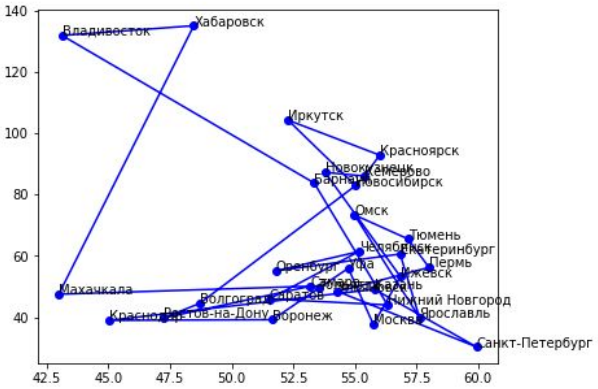





```
Final distance: 26518.1
```

Parameters	T=1e5, iterations=200, a=0.95				
Trials	1	2	3	4	5
Distance	41556.8	41039.9	37527.9	37533.2	43260.8
Time	11.1ms	8.88ms	7.88ms	21.5ms	7.99ms






```
Final distance: 39368.0
```

Comparison

Average results:

	SA T=100 iters=10000 a=0.9999	SA T=1e4 iters=1000 a=0.99	SA T=1e5 iters=200 a=0.95
Distance	19199.3	26435.3	40183.7
Time	309.8ms	42.54ms	11.47ms

As we can see, the greatest alpha, which is annealing rate, equal to 0.9999 gives the best results - 19199.3 on average of 5 runs, still taking less than a second to compute.