

~/warwick-wake-ml

session 2: machine learning for modern astronomy

Thomas Killestein



WarwickAstro/WAKE_workshops

Recap/ feedback

- > Any outstanding technical issues?
- > What did you find challenging?
- > Anything you wanted more of?
- > Hackathon: any ideas?

`./machine-learning-in-astronomy`

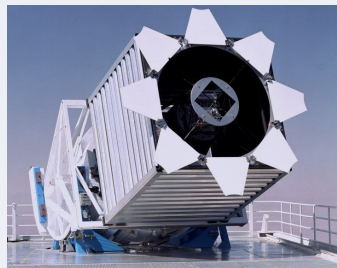
what challenges does astronomy present

machine learning frameworks

prep content for notebooks

Big data in astronomy

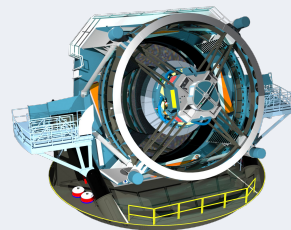
> Astronomy is now among the most data-heavy sciences.



SDSS (1990s)
200GB / night



GOTO (now)
1TB / night



Rubin Observatory
(2024-)
20TB / night



SKA (2030s)
160TB / second

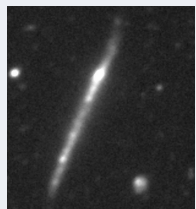
> All of the above make use of machine learning to manage the data volumes involved!

Different data modalities

Images: Spatial



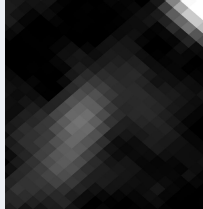
Optical / PS1



NUV / GALEX

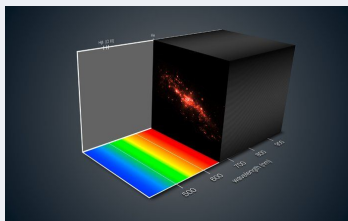
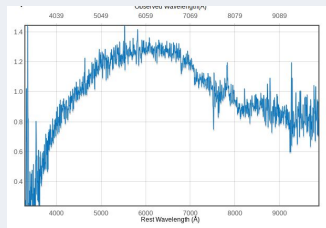


NIR / UKIDSS K

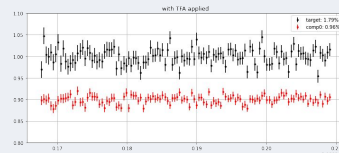
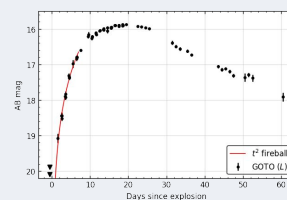


Radio / VLA

Spectra: Wavelength



Light curves: Temporal



Tabular data: Attributes

167.classifier_mzrgrs_8_pkt...	2019-04-08 21:15:00.467000	74.44654464680703	72.8225436
168.classifier_mzrgrs_8_pkt...	2019-04-09 00:00:24.000000	19.38794804488647	20.531516
169.classifier_mzrgrs_8_pkt...	2019-04-09 00:11:55.000000	19.7117772777782	20.531516
170.classifier_mzrgrs_8_pkt...	2019-04-09 01:46:35.333000	18.121360821393	16.4971916
171.classifier_mzrgrs_8_pkt...	2019-04-09 22:00:55.667000	18.94006747071617	18.1689676
172.classifier_mzrgrs_8_pkt...	2019-04-09 23:52:43.533000	18.7028418871408	28.9221885
173.classifier_mzrgrs_8_pkt...	2019-04-09 23:56:37.667000	13.72879792218919	13.8928648
174.classifier_mzrgrs_8_pkt...	2019-04-10 01:30:20.533000	28.2900909799532	31.5164161
175.classifier_mzrgrs_8_pkt...	2019-04-10 01:46:15.667000	18.28129093942968	28.9243981
176.classifier_mzrgrs_8_pkt...	2019-04-11 00:16:22.333000	18.17406037955379	39.45713579
177.classifier_mzrgrs_8_pkt...	2019-04-11 01:15:43.000000	18.4828380992187	11.77524782
178.classifier_mzrgrs_8_pkt...	2019-04-11 01:40:54.667000	16.70538263689385	16.9577437
179.classifier_mzrgrs_8_pkt...	2019-04-13 00:03:06.667000	46.91861214094482	48.18193527
180.classifier_mzrgrs_8_pkt...	2019-04-13 22:10:00.667000	35.97891200220266	54.6629562
181.classifier_mzrgrs_8_pkt...	2019-04-13 23:15:56.667000	46.93828098464339	48.1808752
182.classifier_mzrgrs_8_pkt...	2019-04-15 23:49:55.000000	52.23015487653229	52.46984044
183.classifier_mzrgrs_8_pkt...	2019-04-16 05:21:37.667000	28.4648182395453	35.9208809
184.classifier_mzrgrs_8_pkt...	2019-04-16 05:28:31.667000	38.27490788784427	9.40318406
185.classifier_mzrgrs_8_pkt...	2019-04-16 22:21:32.000000	58.31646708308384	59.948612349
186.classifier_mzrgrs_8_pkt...	2019-04-16 22:25:40.000000	55.18053316151487	56.45749931
187.classifier_mzrgrs_8_pkt...	2019-04-16 22:38:00.000000	50.36083034941229	50.37861303
188.classifier_mzrgrs_8_pkt...	2019-04-16 23:10:18.667000	41.85117828928194	41.80949665
189.classifier_mzrgrs_8_pkt...	2019-04-16 23:50:55.000000	54.41403873772929	54.66884714
190.classifier_mzrgrs_8_pkt...	2019-04-16 00:52:12.667000	54.81306756511264	56.6193868
191.classifier_mzrgrs_8_pkt...	2019-04-16 01:42:01.000000	46.8227786858267	45.9227225
192.classifier_mzrgrs_8_pkt...	2019-04-16 23:42:29.533000	33.48544682878453	52.35932593
193.classifier_mzrgrs_8_pkt...	2019-04-16 23:58:41.333000	46.5883900043216	46.99572956
194.classifier_mzrgrs_8_pkt...	2019-04-20 23:15:56.667000	77.7215105412429	78.14515868
195.classifier_mzrgrs_8_pkt...	2019-04-20 23:16:48.000000	48.447391931147	49.444895404
196.classifier_mzrgrs_8_pkt...	2019-04-24 21:42:32.667000	17.8357079220269	18.2079147
197.classifier_mzrgrs_8_pkt...	2019-04-25 00:19:19.533000	18.18216170291515	16.8962875
198.classifier_mzrgrs_8_pkt...	2019-04-25 01:21:13.000000	15.1540490497775	16.864612
199.classifier_mzrgrs_8_pkt...	2019-04-25 20:59:22.000000	LC_51904256	25.45364079570161
200.classifier_mzrgrs_8_pkt...	2019-04-25 21:26:32.533000	LC_51904256	35.470771328793
201.classifier_mzrgrs_8_pkt...	2019-04-25 21:42:21.000000	LC_51904256	35.4181813084645
202.classifier_mzrgrs_8_pkt...	2019-04-25 21:42:21.000000	LC_51904256	34.4934874183552
203.classifier_mzrgrs_8_pkt...	2019-04-25 21:42:21.000000	LC_51904256	35.52114307840975
204.classifier_mzrgrs_8_pkt...	2019-04-25 22:41:37.667000	LC_51904256	35.454332427193
205.classifier_mzrgrs_8_pkt...	2019-04-25 22:46:48.000000	LC_51904256	36.046464681817
206.classifier_mzrgrs_8_pkt...	2019-04-25 23:53:02.000000	LC_51904256	34.47814644029719
207.classifier_mzrgrs_8_pkt...	2019-04-25 23:57:02.000000	LC_51904256	2.451768726876763
208.classifier_mzrgrs_8_pkt...	2019-04-26 00:24:34.533000	LC_51904256	37.86220912595615
209.classifier_mzrgrs_8_pkt...	2019-04-26 00:51:58.667000	LC_51904256	3.3888897682925
210.classifier_mzrgrs_8_pkt...	2019-04-26 01:11:32.000000	LC_51904256	3.28414914913164
211.classifier_mzrgrs_8_pkt...	2019-04-26 01:26:56.667000	LC_51904256	9.90451097137376
212.classifier_mzrgrs_8_pkt...	2019-04-26 01:59:09.333000	LC_51904256	3.3888897682925
213.classifier_mzrgrs_8_pkt...	2019-04-26 02:12:16.333000	LC_51904256	38.99355346179208
214.classifier_mzrgrs_8_pkt...	2019-04-26 02:17:08.000000	44.53287897810853	43.89427887

Different algorithms for different datasets

- > **Images:** rapidly grow in complexity - regular neural networks become prohibitively expensive relatively quickly. *Convolutional neural networks*
- > **Time-series:** how can we bake causal behaviour and memory into algorithms?
Recurrent neural networks

Time series-style techniques also works for wavelength, language, etc.

./deep-learning

Convolutional neural networks

Instead of using one neuron per input pixel, let's use convolution to create 'feature maps'. Common and well-optimised operation from computer vision!

Filters show strongest 'activation' when neighbourhood of pixels matches the kernel - feature extractors.

Kernel can easily be learned as part of the optimisation process - derivative of convolution is cross-correlation.

Power comes from stacking filters - can very rapidly learn to detect complex features.

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Adaptive gradient descent

> Improve on SGD by:

- > Allowing different model parameters to have different learning rates
- > Changing the global learning rate according to the size of our gradients

adam (Kingma and Ba, 2014)

$$x_{n+1} = x_n - \eta \nabla \mathcal{L}(x_n)$$

Replace gradient with exponentially-weighted average of gradient

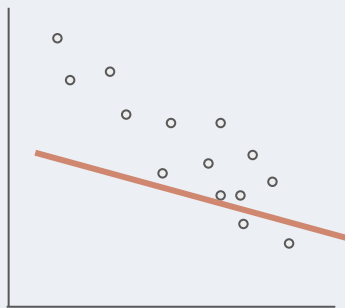
Compute this update element-wise

$$\sqrt{(\nabla \mathcal{L}(x_n))^2 + \epsilon}$$

Scale through by exponentially-weighted RMS of gradient.

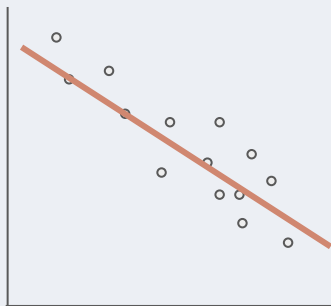
Overfitting

> A central issue in ML is overfitting - where our model simply memorises the input data.



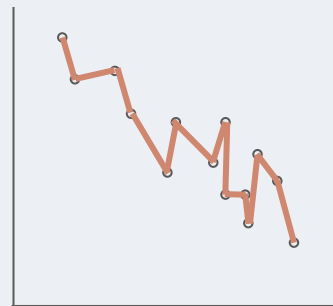
Underfitting:

Model can't represent the data / train harder



'Good fit':

Model learns a sensible (in context of noise) interpretation of data

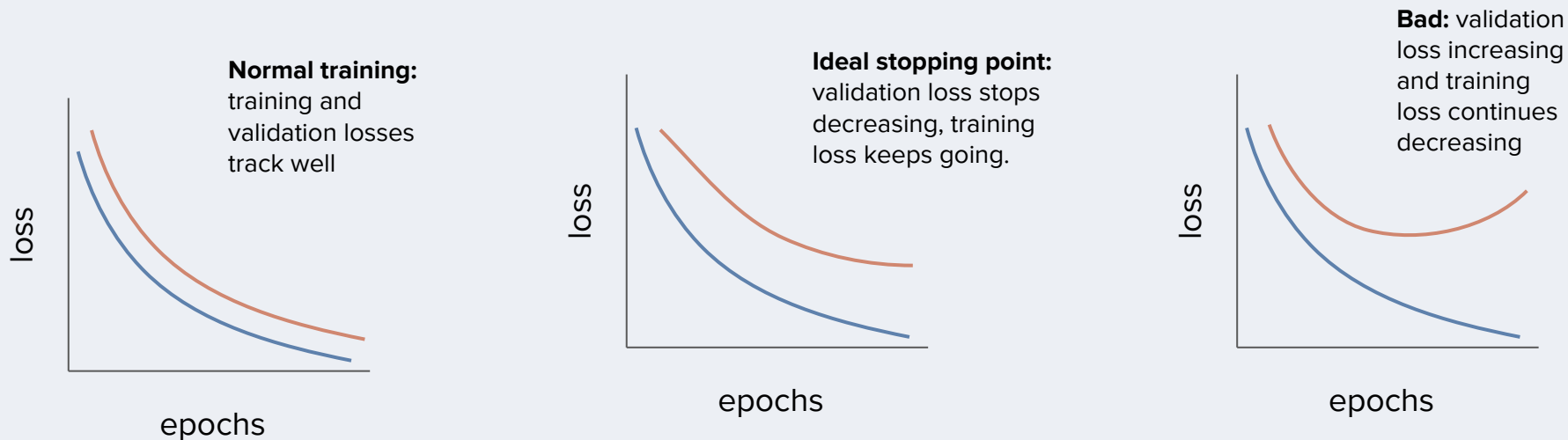


Overfitting:

Model just interpolates between all datapoints, 'learning the noise'

Train, test, validation datasets

> So how to guard against overfitting - hold some data out of the training process to check how your model performs on **unseen data**



> You should also leave a portion of data out entirely, as a 'test' dataset.

Deep learning frameworks

- > Tested and robust implementations of common routines like we wrote yesterday
- > Provide a full end-to-end codebase for rapid prototyping, data preparation, training, evaluating, deploying, and testing deep learning algorithms.
- > There are many - pick your favourite and stick with it. Very similar to 'editor wars'

Usually implement:

Model class

Layers

Optimisers/losses/metrics baked in

Data-loaders and pipelining

Some kind of compilation behind the scenes

JAX Flax

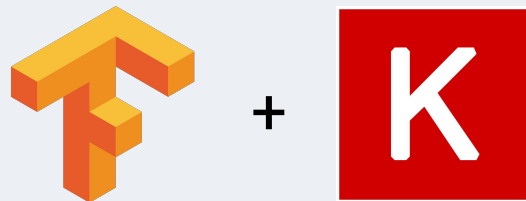
PyTorch

Theano

MXNet

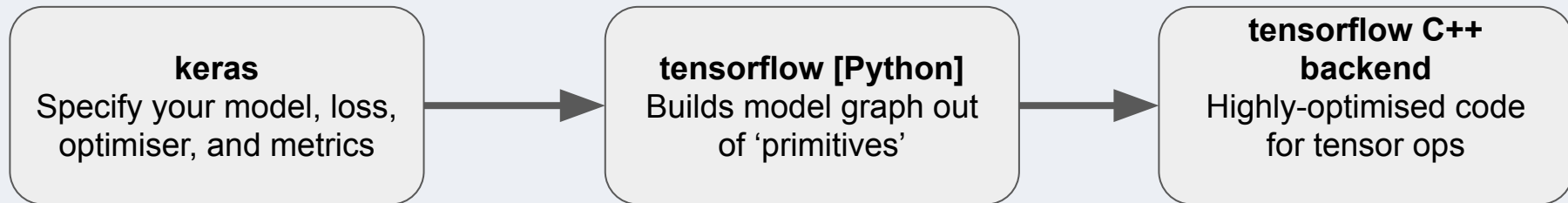
TensorFlow

tensorflow and keras



> tensorflow: low-level differentiable linear algebra on CPU/GPU/TPU - provides gradients, execution flow,

> keras: high-level user-friendly API defining configurable layers, optimisers, metrics, and more.



How to build a keras model

```
model = tf.keras.Sequential(  
    [  
        tf.keras.layers.Something(),  
        tf.keras.layers.SomethingElse(),  
        tf.keras.layers.Something()  
    ]  
)
```

Keras models are initialised as stacks of layers

Layers take data as input, apply some transformation, and yield this as output for the next layer.

Compiling a model associates it with an optimiser, loss, and metrics, and makes it trainable.

```
model.compile(optimiser, loss, metrics=[metric1, metric2])
```

keras models continued

```
history = model.fit(
```

```
    train_data, val_data,
```

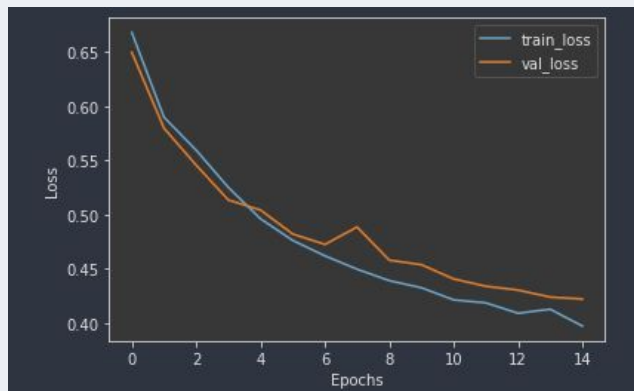
Our data

```
    batch_size, epochs
```

Training parameters

```
)
```

```
history.history ->
```



./over to you!

```
~> git clone git@github.com:WarwickAstro/WAKE_workshops.git  
~> git checkout stable  
~> pip install -r WAKE_workshops/requirements.txt  
~> jupyter lab
```

https://mybinder.org/v2/gh/WarwickAstro/WAKE_workshops/stable