Fastai v2 (2020) [Jeremy Howard, Sylvain Gugger and Rachel Thomas]

fastbook -> fastai book

Lecture1

What you don't need, to do deep learning

Myth (don't need)	Truth
Lots of data	Not true always
Lots of math	High school math is sufficient
Lots of expensive computers	We can get what we need for state of art work for free

AGI -> Artificial General Intelligence

Where is deep learning the best known approach

NLP, CV, Medicine, Biology, Image generation, Recommendation systems, Playing games, Robotics

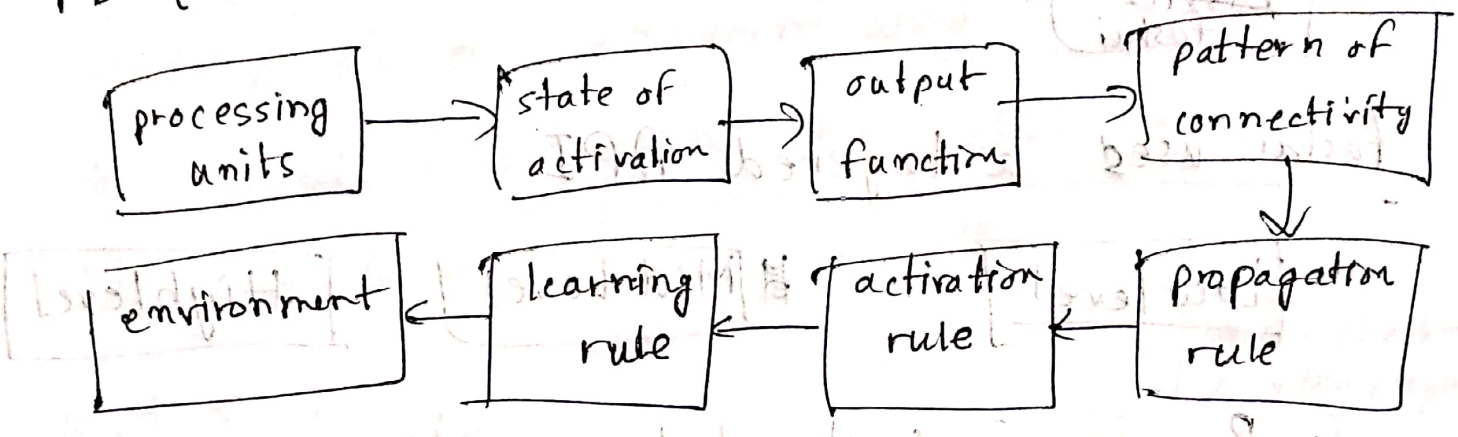
Neural Network: (origin)

1943 Warren McCulloch, a neurophysiologist, and Walter Pitts, a logician, teamed up to develop a mathematical model of an artificial neuron

Mark-I (Frank Rosenblatt)

Marvin Minsky (multilayer pectron) but people ~~misinterp~~ misinterpret

PDP (Parallel Distributed Processing) 1986 MIT



The age of deep learning (1980s)

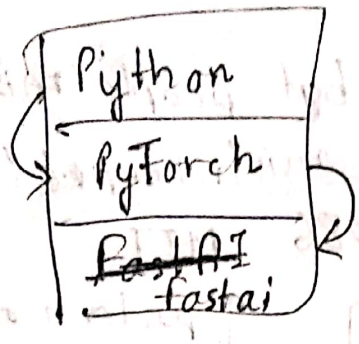
universal approximation theorem: stacking up layers with non-linearity in between can allow any model to ~~learn~~ be approximated

Now we have "a machine capable of perceiving, recognizing and identifying its surroundings without any human training or control."

Course strategy

- Play the whole game
- Make the game worth playing
- Work on the hard parts

[top-down approach]



~~PyTorch~~

fastai used a layered API

Low level | Mid Level | High level

- Removes boilerplate

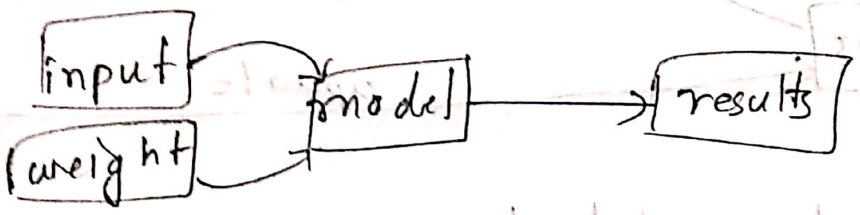
→ Getting a GPU (first learn and do it in colab)

~~Ren~~

Fine-tuning: A transfer learning technique where the params of a pretrained model are update by training for additional epochs → using a different task to that used for pretraining

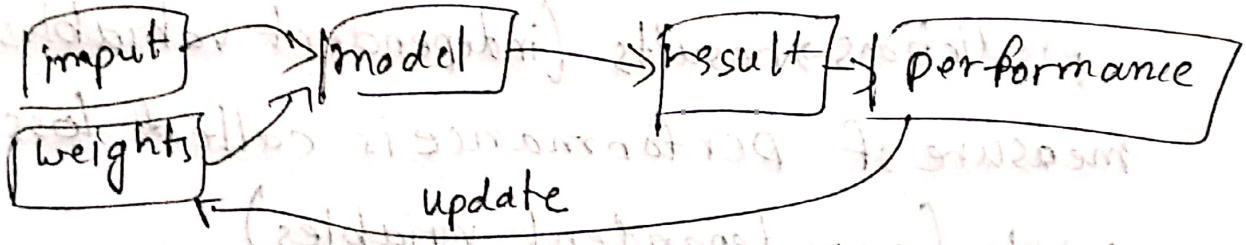
Right back at the dawn of computing in 1949, an IBM researcher named Arthur Samuel started working on a different way to get computers to complete tasks, which he called machine learning. ~~In his classic way 1962~~

1962 →

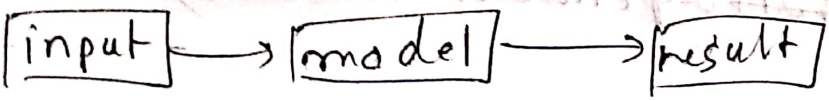


his checkers playing program bebeted Connecticut state champion

Training a machine learning model

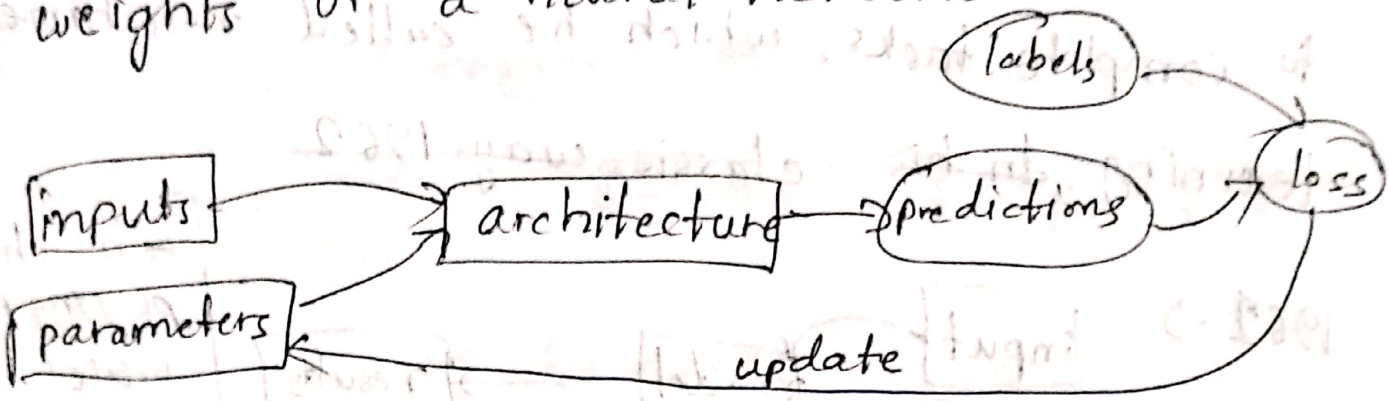


Later



Machine learning: training programs developed by allowing a computer to learn from experience, rather than through manually coding the individual steps.

- SGD (Stochastic Gradient Descent) is a completely general way to update the weights of a neural network



~~para~~ model → architecture

~~p~~weights → parameters

predictions → results (independent variables)

measure of performance is called loss

labels (~~var~~ dependent variables)

Limitations

- ~~Dat~~ need data
- can only learn from input data (creates bias)
- creates predictions → not recommended actions
- Not enough to just have examples of input data we need labels for that data too

Vision, Text, & Tabular, Collaborative filtering

Lecture2

Classification → aims to predict category or class

Regression → attempt to predict numeric quantities

Metric → a function that measures the quality of a model's prediction using validation set.

error_rate accuracy

Overfitting is the single most important and challenging issue. Overfitting is like the model is performing well by cheating.

- check with data that you have need seen.

State of DL

Vision (Detection, Classification)

Text (Classification, Conversation)

Tabular (High cardinality, GPU) Rapids

Recys (Prediction, Recommendation)

Multimodal (Labeling, captioning, Human in the loop)

Other (NLP → Protein)

Catastrophic forgetting → effects after fine-tuning

★ High Temperature and High Humidity Reduce Transmission of COVID-19

How might we decide if there's a relationship?

- Pick a "null" hypothesis
- Gather data of independent & dependent variables
- What % of time would we see that relationship by chance?

→ Never do null hypothesis or p-value testing

DataBlock API

get_items: →

Vision, Text, Tabular, Collaborative filtering
(segmentation)

Lecture2

about fine-tune

Classification → aims to predict category or class

Regression → attempt to predict numeric quantities

Metric → a function that measures the quality of a model's prediction using validation set.

error_rate accuracy

Importance of validation set:

Overfitting is the single most important and challenging issue. Overfitting is like the model is performing well by cheating

- check with data that you have need seen.

Transfer learning: Using a pretrained model for a different task to what it was originally trained for

State of DL

Vision (Detection, Classification)

Text (Classification, Conversation)

Tabular (High cardinality, GPU) Rapids

Recys (Prediction, Recommendation)

Multimodal (Labeling, captioning, Human in the loop)

Other (NLP → Protein)

fine-tune:

why does transfer learning work? (image)

- deep layer capture and learn structures

Catastrophic forgetting → effects after fine-tuning

★ High Temperature and High Humidity Reduce Transmission of COVID-19 (proven false by Jeremy) (by reverse engineering)

How might we decide if there's a relationship?

- Pick a "null" hypothesis"
- Gather data of independent & dependent variables
- What % of time would we see that relationship by chance?

→ Never do null hypothesis or p-value testing (6 Concepts from statistics)

~~DataBlock API~~

~~get_items:~~

Turn predictive model into something useful in production

Strategy → Data → Analytics → Implementation → Maintainance

Identify and manage constraints

- Designing great data products (JHoward et.al.)

DataBlock API

get_items

- get_x (getter)
- get_y

- block[0].create
- block[1].create

item_tfms (transforms the items)

collate (DataLoader) — PyTorch DataLoader for better GPU performance

batch_tfms (batch transforms)

Lecture 3 | Lesson 3

Day#3

Resizing Images with DataBlock

Squishing → Distorts the image

Padding → ~~Misus~~ Lots of misused computations

RandomResizeCrop → Most practical (Crop and zoom randomly)

Crop → Lose information

→ solves overfitting as well (Data augmentation)

Data Augmentation:

Synthesizing new data from existing data

Batch_tfms → allows you to do augmentation in GPU

- flipping
- flipping vertically
- max_rotate
- max_zoom
- max_lighting
- max_warp

and so forth

- fastai will automatically avoid data augmentation in validation set
- we can write our own transforms and use it with fastai if need (Python, PyTorch code)

Data Cleaning

- ImageClassifier Cleaner (allows us to choose a category and training vs validation se and others)
- fastai is implemented on notebooks

Turning model into an online app: (by exporting them in a pickle file)

How to avoid disaster in deployment?

↳ Model Bias (the data we are training with model can have different type of mix in real life

- Building ML powered apps → Going from idea to product

Problems with data

- out-of-domain data (have different distribution than what we train with)
- domain shift

Train on yesterdays data

Fine-tune on todays data

Getting out of ~~disasters~~

Manual process

↳ Limited scope deployment

↳ Gradual Expansion

Unforeseen consequences

Feedback loops (take a minor problem and turns it into a big issue)

example → predictive policing

04_mnist_basics:

Training a digit classifier

MNIST ⟷ LeNet-5 (Yann Lecun et al)

path.ls() ⇐ like linux ls(ls)

fastcore → doesn't come from PyTorch and pandas and heavy libraries like this (new addition in fastai)

?? → where it comes from doc(...) ⇐ shows the documentation

?? → source code

Classification analogy

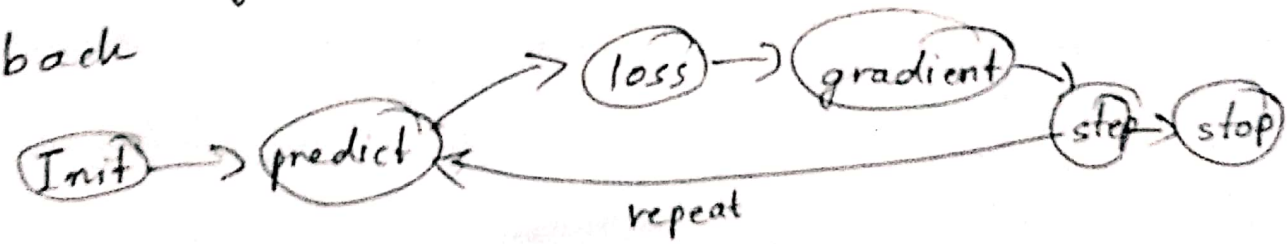
Baseline → simple model in which we are confident and perform reasonably well

Pixel similarity

- list and distionary comprehensions are wonderful feature of python
- Broad casting

Stochastic Gradient Descent (SGD):

- Initialize weight
- For each sample use these weights to predict
- Based on those predictions, calculate how good the model is (loss)
- Calculate the gradient
- Step all weights based on that calculation
- Go back



- goal - is to get to the global minima

The gradient

Automatic ~~differenti~~ differentiation → PyTorch

backward()

grad

Stepping with learning rate:

w -= gradient(w) * lr