Lec 11 Distillation, Pruning, Quantisation

Today: training model expensive, deployment more expensive
models getting bigger

Model compression:
- quantisation — less bits
- pruning — remove some components
- distillation — train small model to do same thing

Intuition

Why not start with small model? Why is it possible to compress
- Overparametrised models easier to optimise, in theory
  - easier to find optima in non-convex space, e.g. side step at saddle points
- At inference, we don't need those training tricks

Quantisation

▷ Post-training quantisation

* Size in mem linear to bit per param

Float:	sign bit	exponent	fraction
float16		- 5 bit	10 bit
bfloat16		- 8 bit	7 bit

▷ Int8 absolute max quantisation

Take absmax of vector, scale everything to the i8 range
  └ (with some rounding)

→ Model-Aware quantisation (GOBO)
- look at distribution of the weights, then decide how to quantise with minimal info loss
  └ per layer
- store outliers in dist. in full precision, i8 compress others

→ LLM.int8
- like previous, but quantise per row/col of matrix instead of over whole layer

Overhead: encode & decode the numbers
for bigger models this could double the speed

Hardware constraints: only some types are supported
- eg. not int3, no int4 in PyTorch either
so quantisation may require low level code

▷ Quantisation-aware training

1 bit precision won't work post training, but can work if trained on 1 bit, with fancy statistics

▷ Layer-by-Layer Quantisation-aware Distillation

For one layer at a time, train a quantised version to mimic the original model at that step

▷ Q-LORA ← popular

Unstructured Pruning ← remove some params

▷ Magnitude pruning

Set some % of least magnitude model to 0, as they don't do much

▷ Lottery ticket hypothesis — some subnetwork of randomly initialised model can work better than the whole model

So prune then train

▷ Wanda (CMU, 2023)

Problem with magnitude pruning: doesn't consider input magnitude so small param processing large input can get pruned to 0.

Sol: look at input

* Problems with unstructured pruning: maybe no hardware opt for sparse matrices

Structured Pruning ← remove components

→ Remove half of Transformer's heads

→ Mask out some components, learn which to mask out
  └ but expensive to train

→ Prune using forward pass. Randomly mask things out & test, use regression model on result to decide what to prune

→ Do bayesian search?

Distillation

Train a model to replicate another model
  └ all param can be different, so is architecture

* Weak supervision: use pseudolabels

→ Self-training – model make own train data
→ Co-training
→ Meta pseudo-labels
→ Rule-based heuristic to make pseudolabels

Hard target: match the label from teacher model
Soft target: match the probability distribution from teacher model

→ Born Again NN — repeatedly distill model to itself using soft target makes the model better. essentially ensembling many disturbed versions of itself

* Deep NNs are usually robust to label noise
  at least uniformly sampled

Sequence-Level Distillation

→ Match teacher at each point in distribution process
→ If teacher & diverge when generating, that's bad. So also generate hard label from teacher

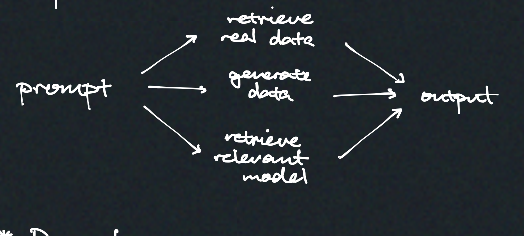
DistillBERT
- Every 2 layer into 1 layer, by initialising as one of the layers then soft target training
- Cosine similarity btwn student & teacher hiddens
- Both supervised & distillation-based loss

Self-Instruct
- Make pseudolabels for self, do instruction fine tuning
- Trick: generate input based on label, rather than the other way around.

* Idea: Assymetry

$$X \underset{\text{easy}}{\overset{\text{hard}}{\rightleftarrows}} Y$$
← generate data in this direction

Prompt2Model



* Recent:
concept of distillation shifting towards Synthetic Data Generation