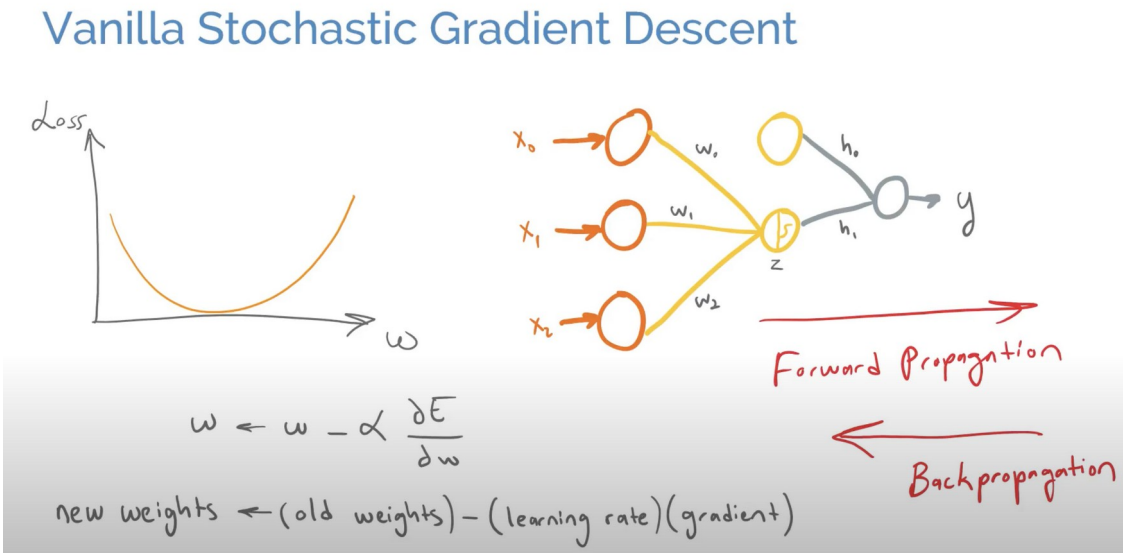
```python
import gradio as gr

from transformers import pipeline

pipe = pipeline("translation", model="Helsinki-NLP/opus-mt-en-es")

def predict(text):
    return pipe(text)[0]["translation_text"]

iface = gr.Interface(
  fn=predict,
  inputs='text',
  outputs='text',
  examples=[[""]]
)

iface.launch()
```



Momentum

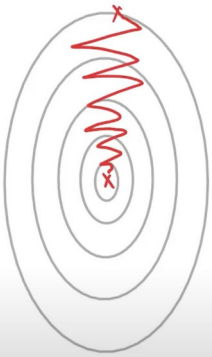
(new weights) <- (old weights) – (learning rate) (gradient)
+ past gradients

weight of past gradients

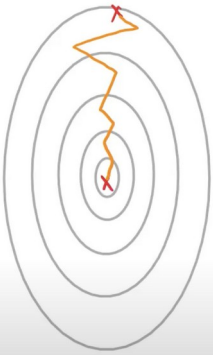
(accumulator) <- (old accumulator) (momentum) + (gradient)

(new weights) <- (old weights) – (learning rate) (accumulator)

Momentum



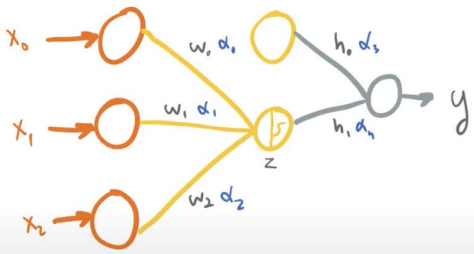
Stochastic Gradient Descent



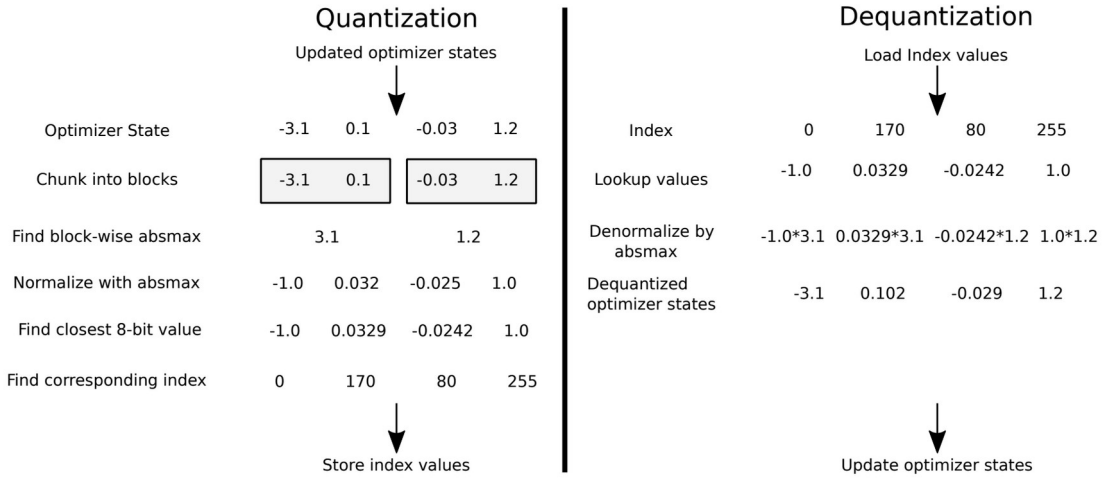
Stochastic Gradient Descent
with Momentum

Adaptive Methods

- AdaGrad
  - Large gradient: decrease α faster
  - Small gradient: decrease α slower
- RMSProp
  - Moving average of gradients
- Adam
  - Adaptive Moment Estimation
  - RMSProp + Momentum

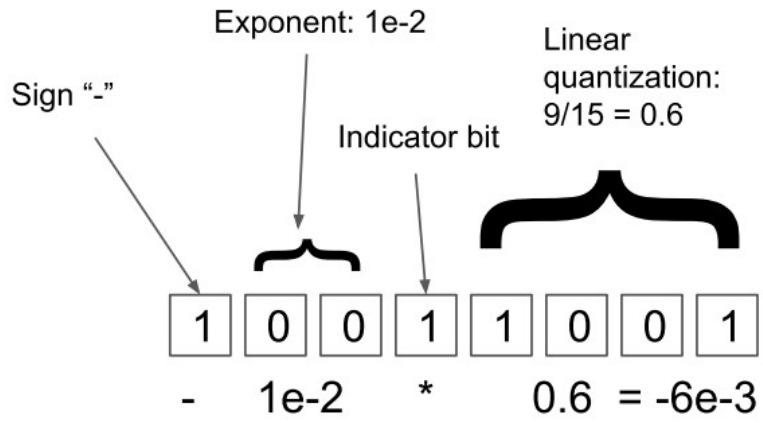


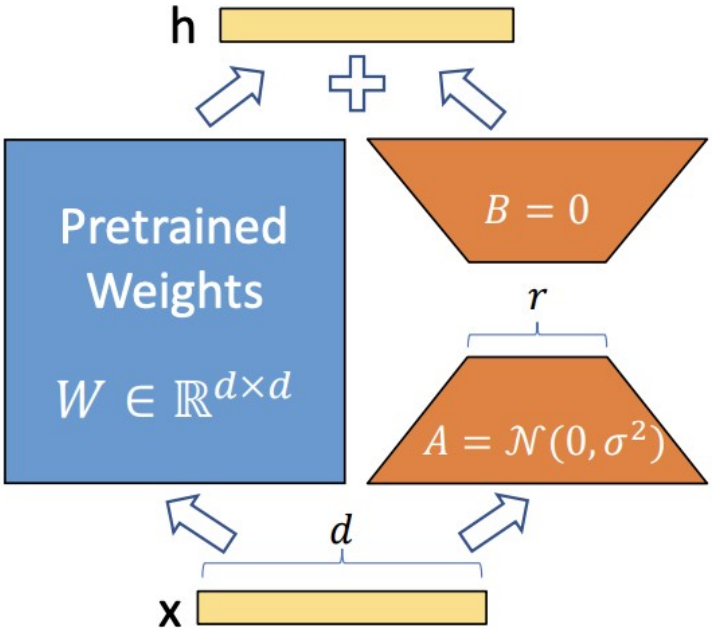
- Quantization is the process of mapping continuous infinite values to a smaller set of discrete finite values.
- The process of casting encodings of categorical data from the discrete space to continuous space is called dequantization



$$\text{Momentum}(\mathbf{g}_t, \mathbf{w}_{t-1}, \mathbf{m}_{t-1}) = \begin{cases} \mathbf{m}_0 = \mathbf{g}_0 & \text{Initialization} \\ \mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + \mathbf{g}_t & \text{State 1 update} \\ \mathbf{w}_t = \mathbf{w}_{t-1} - \alpha \cdot \mathbf{m}_t & \text{Weight update} \end{cases}$$

$$\text{Adam}(\mathbf{g}_t, \mathbf{w}_{t-1}, \mathbf{m}_{t-1}, \mathbf{r}_{t-1}) = \begin{cases} \mathbf{r}_0 = \mathbf{m}_0 = \mathbf{0} & \text{Initialization} \\ \mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t & \text{State 1 update} \\ \mathbf{r}_t = \beta_2 \mathbf{r}_{t-1} + (1-\beta_2)\mathbf{g}_t^2 & \text{State 2 update} \\ \mathbf{w}_t = \mathbf{w}_{t-1} - \alpha \cdot \frac{\mathbf{m}_t}{\sqrt{\mathbf{r}_t}+\epsilon} & \text{Weight update,} \end{cases}$$





Links:

- https://arxiv.org/pdf/2110.02861.pdf
- https://arxiv.org/pdf/2106.09685.pdf
- https://github.com/TimDettmers/bitsandbytes
- https://colab.research.google.com/drive/1ft6wQU0BhqG5PRlwgaZJv2VukKKjU4Es?usp=sharing
- https://www.forefront.ai/blog-posts/how-to-fine-tune-gpt-j
- https://github.com/kingoflolz/mesh-transformer-jax/
- https://pile.eleuther.ai/

- https://www.machinelearningnuggets.com/gradio-tutorial/
- https://gmihaila.medium.com/fine-tune-transformers-in-pytorch-using-transformers-57b40450635
- https://colab.research.google.com/github/DerwenAI/spaCy_tuTorial/blob/master/BERT_Fine_Tuning.ipynb

Homework (final project):

- finetune GPT-J or other BLM on gathered dataset (it may be any of used earlier datasets or gathered)
- create a flask or gradio application that takes a url or text as an input (if a url it should parse a text out of it)
- checks if it is a fake news or not
- summarize it
- produce list of topics containing in the text
- wrap in Docker container to have an ability to use it