

Sep 07, 22 14:49	main.py	Page 1/2
<pre>#!/bin/env python3.8 """ Example assignment. Author: Chris Curro """ import os # from turtle import shape import matplotlib.pyplot as plt import numpy as np import tensorflow as tf import math from tqdm import trange script_path = os.path.dirname(os.path.realpath(__file__)) rng = np.random.default_rng(seed=42) class Data: def __init__(self, ns, sig, range): self.rng = np.random.default_rng(seed=42) self.index = np.arange(ns) self.num_samples = ns self.sigma = sig self.range = range self.x = np.array(self.rng.uniform(0, 2, self.num_samples)) clean_y = tf.math.sin(2 * math.pi * self.x) self.y = np.array(clean_y + self.rng.normal(0, self.sigma)) def get_batch(self, batch_size): choices = np.array(rng.choice(self.index, size=batch_size)) return self.x[choices], self.y[choices].flatten() class Model(tf.Module): def __init__(self, M): self.M = M self.w = tf.Variable(rng.normal(0, 1, M)) self.mu = tf.Variable(rng.normal(1, 0.5, M)) self.sigma = tf.Variable(rng.normal(0, 1, M)) self.b = tf.Variable(np.array([[0]])) def __call__(self, x): print(self.mu) print(self.w) y_hat = 0 for j in range(self.M): theta_j = tf.math.exp(-((x - self.mu[j]) ** 2) / (self.sigma[j] ** 2) y_hat += tf.cast(theta_j * self.w[j], dtype="float32") + tf.cast(self.b, dtype="float32") return y_hat data = Data(50, 0.1, (0, 2)) model = Model(5) optimizer = tf.optimizers.SGD(learning_rate=0.1)</pre>		

Sep 07, 22 14:49	main.py	Page 2/2
<pre>bar = trange(500) for i in bar: with tf.GradientTape() as tape: x, y = data.get_batch(16) y_hat = model(x) loss = 0.5 * tf.reduce_mean((y_hat - y) ** 2) grads = tape.gradient(loss, model.trainable_variables) optimizer.apply_gradients(zip(grads, model.trainable_variables)) bar.set_description(f"Loss @ {i} => {loss.numpy():0.6f}") bar.refresh() fig, ax = plt.subplots(1, 2, figsize=(11, 4), dpi=200) ax[0].set_title("Linear Combination of Gaussians") ax[0].set_xlabel("x") ax[0].set_ylim(np.amin(data.y) * 1.5, np.amax(data.y) * 1.5) h = ax[0].set_ylabel("y", labelpad=10) h.set_rotation(0) xs = np.linspace(0, 2, 100) xs = xs[:, np.newaxis] ax[0].plot(xs, np.squeeze(model(xs)), dashes=[6, 2], label="model") ax[0].plot(np.squeeze(data.x), data.y, "o", label="training data") ax[0].plot(xs, np.squeeze((tf.math.sin(2 * math.pi * xs))), label="training data") ax[0].plot() ax[1].set_title("Linear Combination of Gaussians") ax[1].set_xlabel("x") ax[1].set_ylim(np.amin(data.y) * 1.5, np.amax(data.y) * 1.5) for mu_i in range(tf.shape(model.mu)[0]): theta_j = tf.math.exp(-((xs - model.mu[mu_i]) ** 2) / (model.sigma[mu_i] ** 2) ax[1].plot(xs, theta_j) ax[0].plot() ax[1].plot() plt.tight_layout() plt.savefig(f"{script_path}/fit.pdf")</pre>		