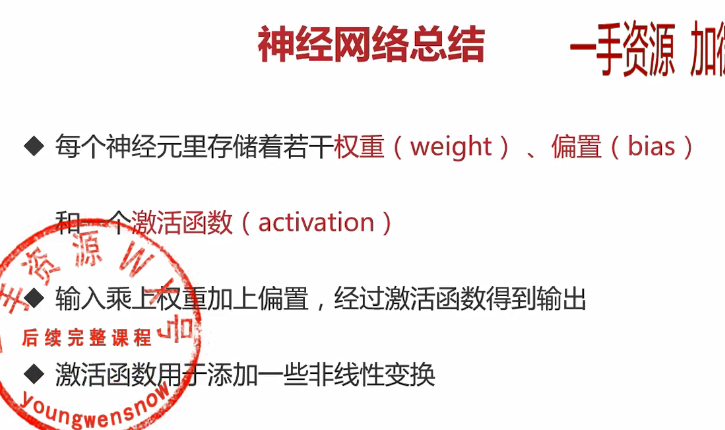
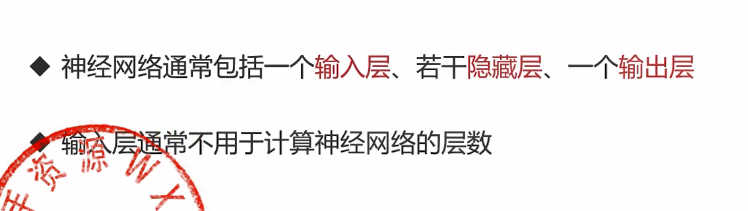
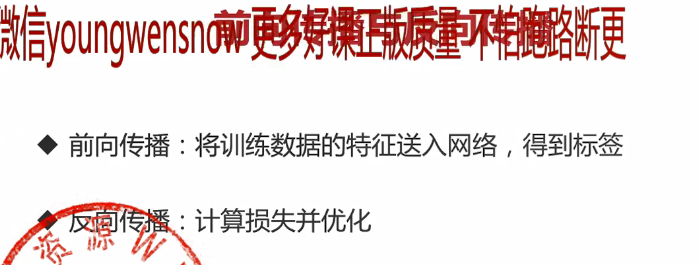
神经网络：





前向传播与反向传播



在浏览器上安装

<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js"></script>

npm install @tensorflow/tfjs

张量（tensor）：张量是向量和矩阵向更高维度的推广，神经网络每一层要存储n维数据，所以要用tensor

相当于多维数组

const a =tf.tensor(1);

点乘.dot

例

tf.tensor([[1,2,3,4],[2,3,4,5],[3,4,5,6],[4,5,6,7]]).dot(tf.tensor([1,2,3,4])).print();

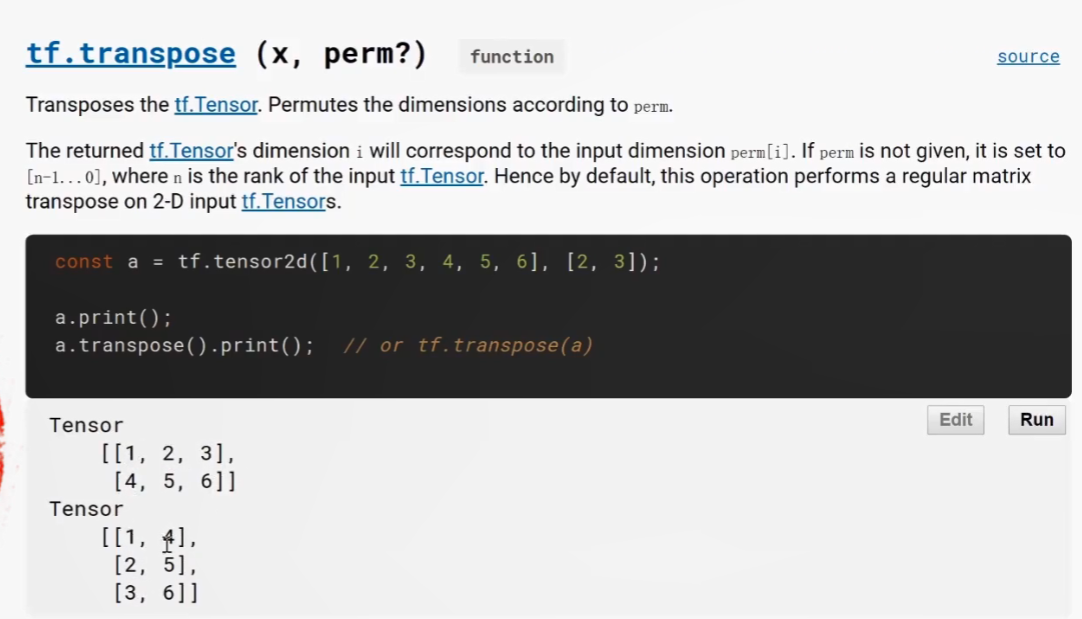
输出

Tensor

[30, 40, 50, 60]

后面的tensor每个数与前面数组依次相乘，前面是权重，后面是输入

反转.transpose



线性回归

tf可视化工具

安装：

npm i @tensorflow/tfjs-vis -S

引用：

import \* as tfvis from '@tensorflow/tfjs-vis'

用法：

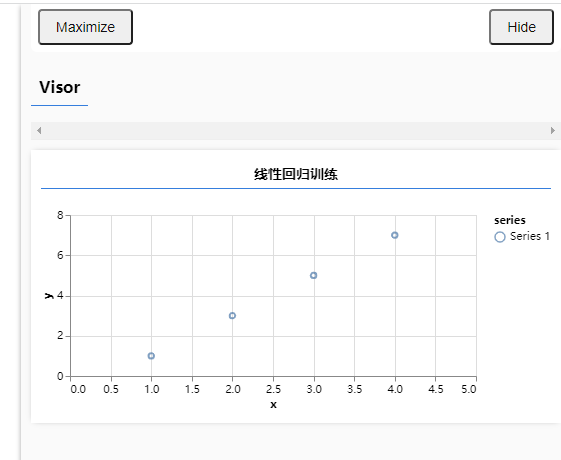
tfvis.render.scatterplot(

{name:'线性回归训练'},

{values:xs.map((x,i)=>({x,y:ys[i]}))},

{xAxisDomain:[0,5],yAxisDomain:[0,8]},//控制边界

);



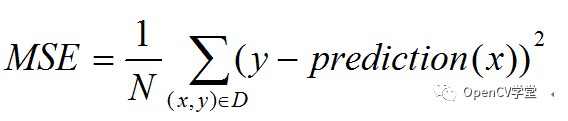
创建连续的模型

const model= tf.sequential();

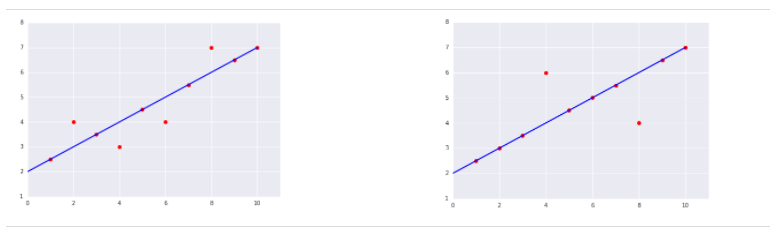
增加神经元：

model.add(tf.layers.dense({units:1,inputShape:[1]}));

均方误差：



请求出各个样本的所有平方损失之和，然后除以样本数量



右边均方误差高

给模型添加损失函数：

model.compile({loss:tf.losses.meanSquaredError});//均方误差损失函数

增加优化器（随机梯度下降法）

model.compile({loss:tf.losses.meanSquaredError,optimizer:tf.train.sgd(0.1)});

讲数据变成tensor

const inputs = tf.tensor(xs);

const labels = tf.tensor(ys);

模型拟合，因为是异步，需要async await

    const inputs = tf.tensor(xs);

    const labels = tf.tensor(ys);

    await model.fit(inputs,labels,{

        batchSize:1,//每次训练的批量

        epochs:100,//迭代次数

        callbacks:tfvis.show.fitCallbacks(

            {name:'训练过程'},

            ['loss']

            )//可视化训练过程

    });

应用模型：

    const output = model.predict(tf.tensor([50]));

    alert(`如果 x 为 5，那么预测 y 为 ${output.dataSync()[0]}`);

归一化：将大数量级变成小数量级，最好是[0,1][-1,1]

为什么要归一化：绝大部分tensorflowjs的模型都不是给特别大的数设计的

将不同数量级的特征转换到同一数量级，防止某个特征影响过大

归一化练习：

    const heights = [150,160,170];

    const weights = [40,50,60]

    tfvis.render.scatterplot(

        {name:'身高体重数据'},

        {values:heights.map((x,i)=>({x,y:weights[i]}))},

        {xAxisDomain:[140,180],yAxisDomain:[30,70]}

    );

    const inputs = tf.tensor(heights).sub(150).div(20);//sub减，div除

    const labels = tf.tensor(weights).sub(40).div(20);

先将输入数据进行归一化，本案例是减去最小值再除以长度

   const output = model.predict(tf.tensor([180]).sub(150).div(20));

    alert(`如果 身高 为 180，那么预测 体重 为 ${output.mul(20).add(40).dataSync()[0]}kg`);//mul乘，add加

同时运用模型时也要归一化，输出的结果要反归一化

.dataSync()[0]}

将tensor数据转化成普通数据

逻辑回归