深度学习与中文自然语言处理

Deep Learning for NLP based on TensorFlow

April-29 2018

Outlines

- 1. Review and talk for assignment and project;
- 2. Neural Networks and Deep Learning First Step;
- 3. Mini-Tensorflow

Question

- 基于规则的有什么缺点：
- [ ] A. 规则复杂， 规则直接可能互相冲突
- [ ] B. 不能及时适应新的问题
- [ ] C. 需要反复修改
- [ ] D. 向别人解释困难

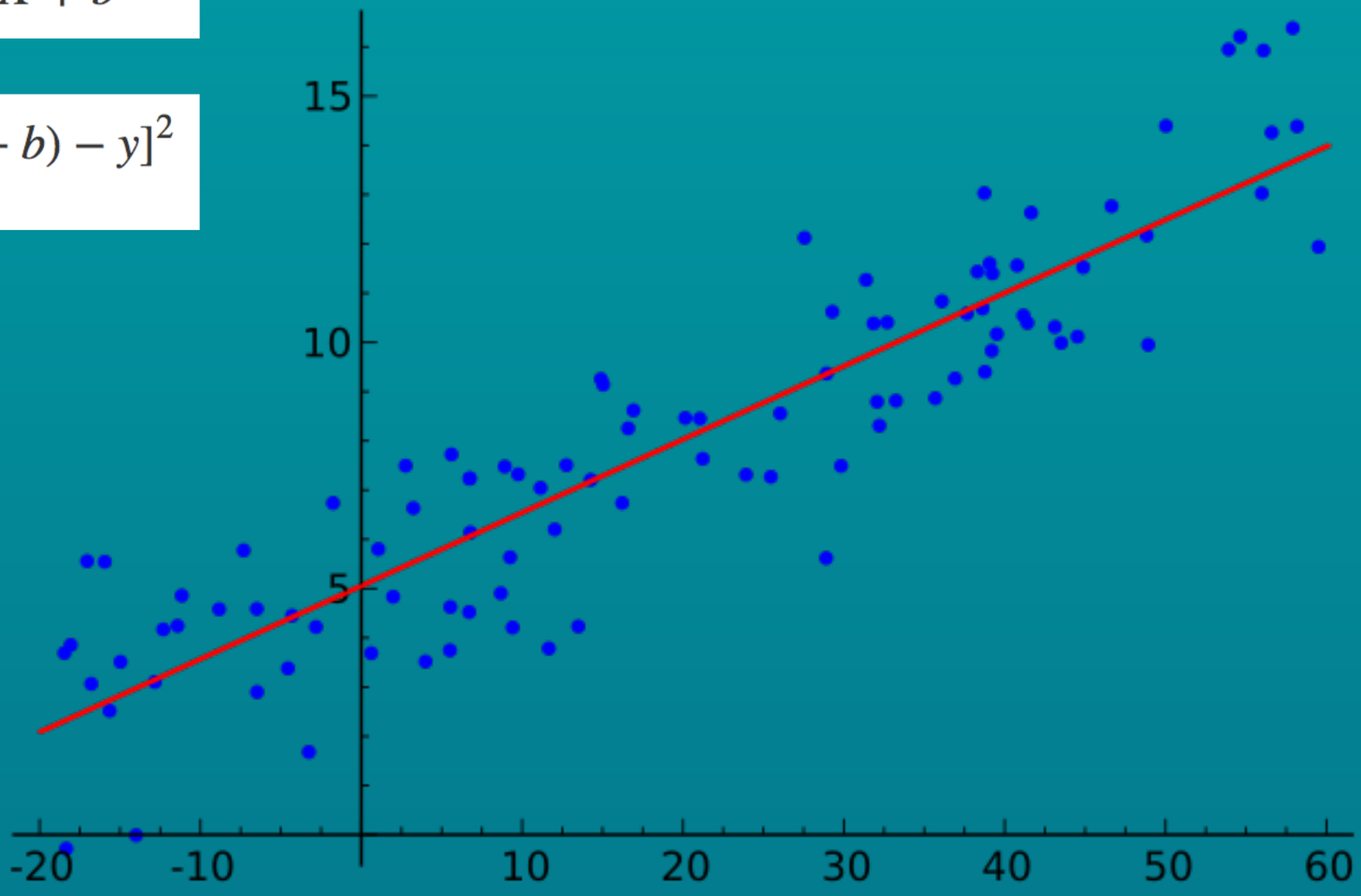
Review

- 1. 什么是机器学习
- 2. 机器学习与基于人工规则的区别， 举例子说明;
- 3. 机器学习有什么类型
- 4. 欠拟合与过拟合

一个例子

$$\hat{Y} = aX + b$$

$$\sum_{x \in X} [(ax + b) - y]^2$$



我们发现了什么?

- 1. 利用计算机反复迭代自动调节；(Train Epoch)
- 2. 给计算机设定目标; (Loss Function)
- 3. 设定更新策略； (Optimizer)
- 4. 使学习出来的参数比较“均匀”; (Regularization)

机器学习工作者的目标

- 1. 建立一个程序(模型, Model)， 使得Loss函数能够降低至0；
- 2. 降低至0之后， 能够在其他未曾见过的数据集上取得同样小的Loss
- 3. 能够缩短训练时间；
- 4. 如何使用小样本进行训练；
- 5. 以及其他（表示学习， active learning）

机器学习的类型

- 1. Classification
- 2. Regression
- 3. Policy & Optimization （Reinforcement Learning）
- 4. Cluster （K-means）
- 5. Generative Model （GAN）

神经网络的基本原理

- 输入： X
  - X： <x1, x2, … ,xN>
- 输出： Y
  - Y： 0.3, 1.4 等数值
  - Y: [0, 0, 1], [1, 0, 0], [0, 1, 0]等one-hot类型

如何实现?

- 假设X和Y之间满足某种函数关系:
  - Y = f(X)
- Y如何表示?
-

$$f(x) = ax + b$$
$$f(x) = sigmoid(ax + b)$$
$$f(x) = ax^2 + bx + c$$
$$f(x) = sin(x) + cos(x)$$

- 非线性变换：

$$f(x) = NoLinear_1(a_1 * NoLinear_0(a_0 x + b_0) + b_1)$$

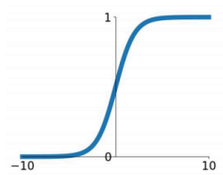
- 非线性变换能力的叠加：　（层数 Layer）

- a1, a2：Weights 权重

- 常用的非线性函数：　（activation Function 激活函数）

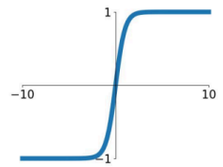
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



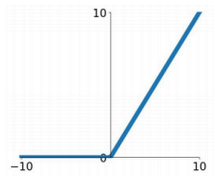
tanh

$$\tanh(x)$$



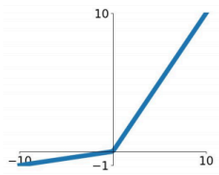
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

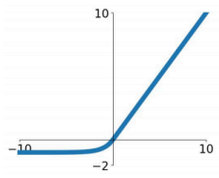


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

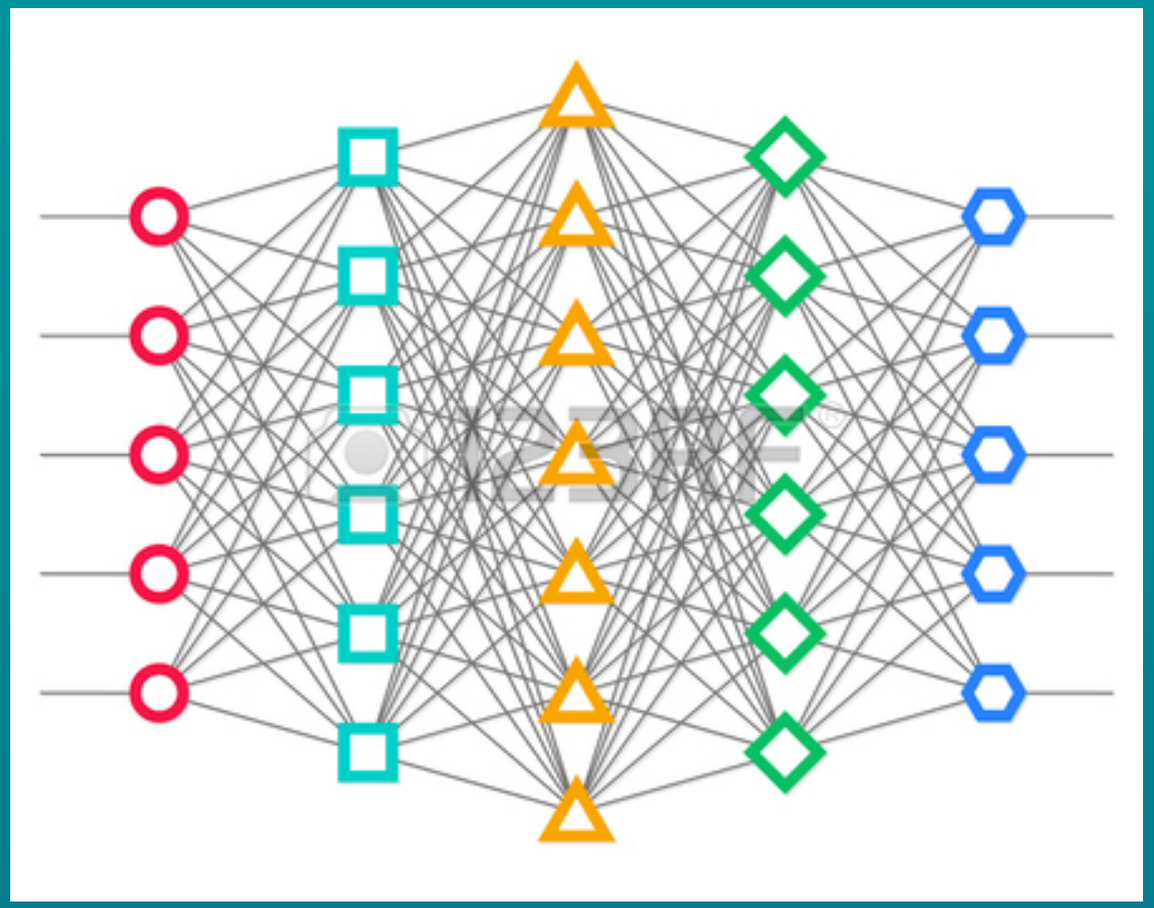
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



图像化表示

- 输入： X = [x0, x1, x2, … Xn]



AlexNet

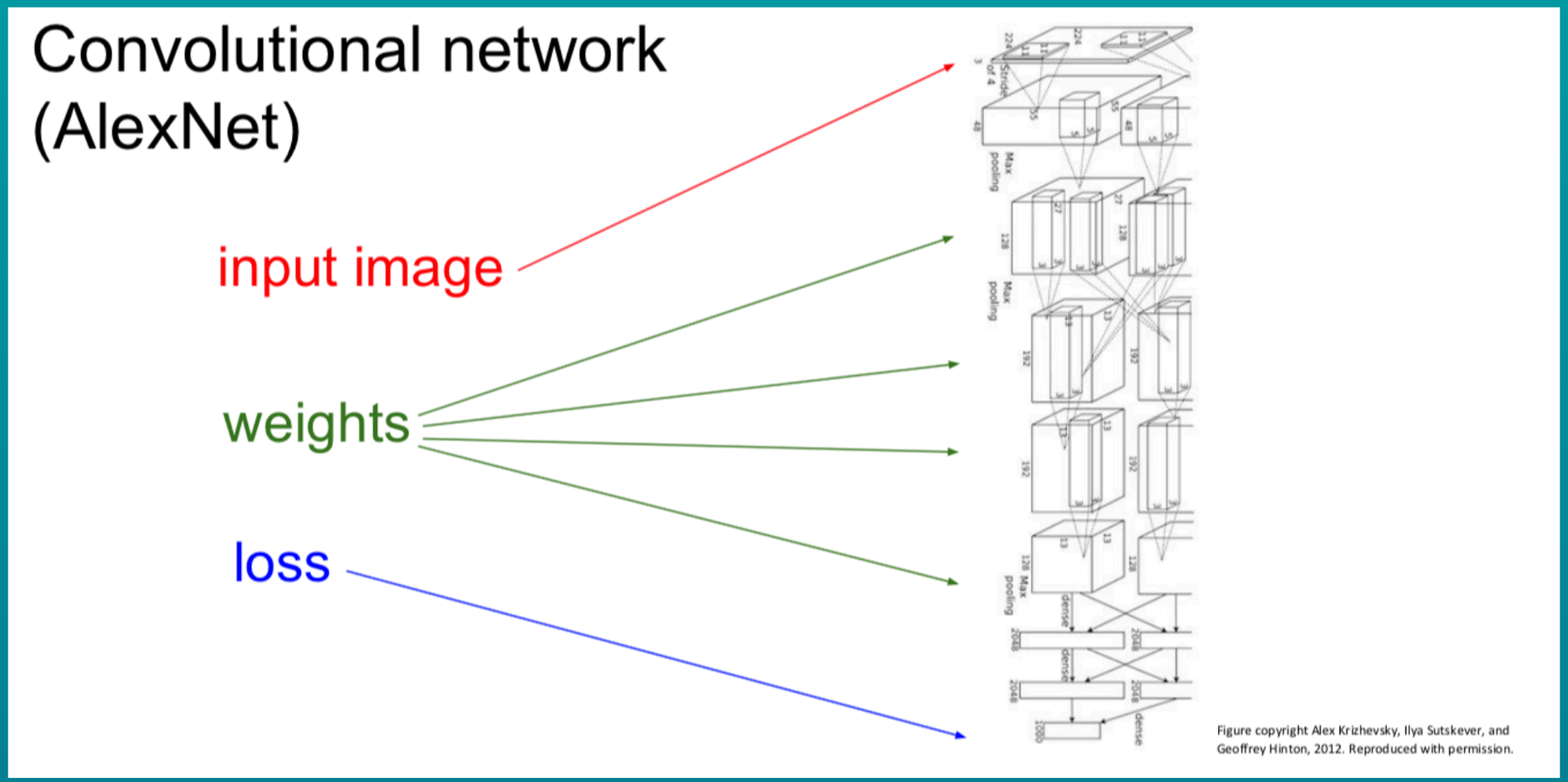


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Loss函数

- 1. Regression问题： Loss 定义为：

$$Loss(f(\mathbf{x}); \mathbf{y}) = \sum_{i=1}^{M} (y_i - f(x_i))^2$$

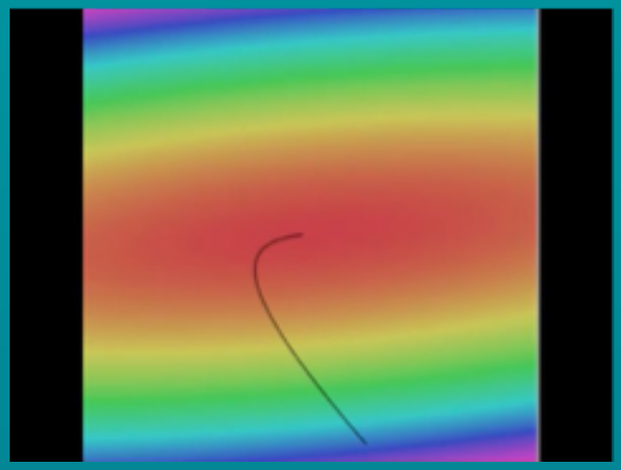
$$f(x) = NoLinear_1(a_1 * NoLinear_0(a_0 x + b_0) + b_1)$$

- 2. 问题定义为： 找到一组a0, a1, b0, b1, .. aN, bN 使得Loss最小
  - *But How?*

示意图

Backpropagation

- 梯度是什么?



$\nabla\varphi$或$\operatorname{grad}\varphi$

其中∇（nabla）表示向量微分算子。

$\nabla\varphi$在三维直角坐标中表示为

$$\nabla\varphi = \left(\frac{\partial\varphi}{\partial x}, \frac{\partial\varphi}{\partial y}, \frac{\partial\varphi}{\partial z}\right)$$

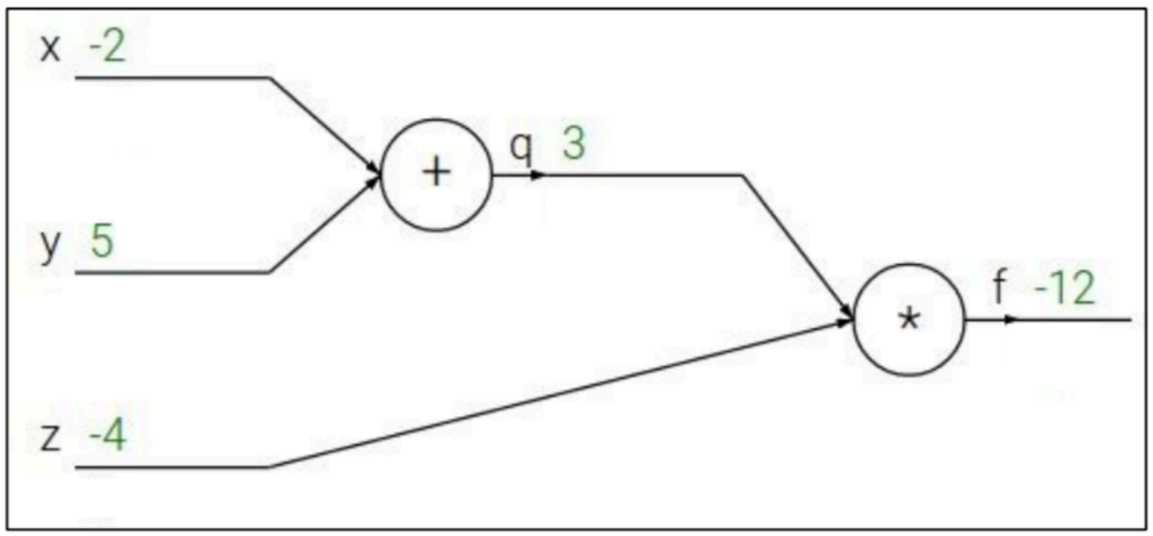
```
while True:
    weight_grad = evaluate_gradient(loss_fun, data, weights)
    weights += - step_size * weights_grad
```

Backpropagation

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} \qquad \frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Image: Stanford CS231n Deep Learning For Computer Vision

Loss函数

- 1. 对于Regression 回归预测问题：
  - SSE

$$Loss(f(\mathbf{x}); \mathbf{y}) = \sum_{i=1}^{M} (y_i - f(x_i))^2$$

- 2. 对于分类问题：
  - softmax + cross entropy

示意图

Softmax

- 需要预测的值为一个类型： [0， 0， 0， 1]
- 经过计算， 模型为每一个标签打分： [1.3, -1.3, 0.2, 2.5]

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, ..., K.$$

```
y_hat = [1.3, -1.3, 0.2, 2.5]
y_hat = np.exp(np.array(y_hat)) / sum(np.exp(np.array(y_hat)))
y_hat
```

```
array([ 0.21153896,  0.01571176,  0.0704152 ,  0.70233408])
```

将每个预测值变为概率的形式

Softmax

- 获得了每个类型预测的概率
- 我们的目标： 正确的标签的值远远大于其他， 最好为1.0
- 如何进行： 交叉熵

Cross Entropy

- 熵： 物体的混乱程度
- 信息熵:

$$\mathrm{H}(X) = \sum_{i=1}^{n} \mathrm{P}(x_i)\,\mathrm{I}(x_i) = -\sum_{i=1}^{n} \mathrm{P}(x_i)\log_b \mathrm{P}(x_i),$$

$$-\sum_{c \in C} y_c * log(\hat{y_c})$$

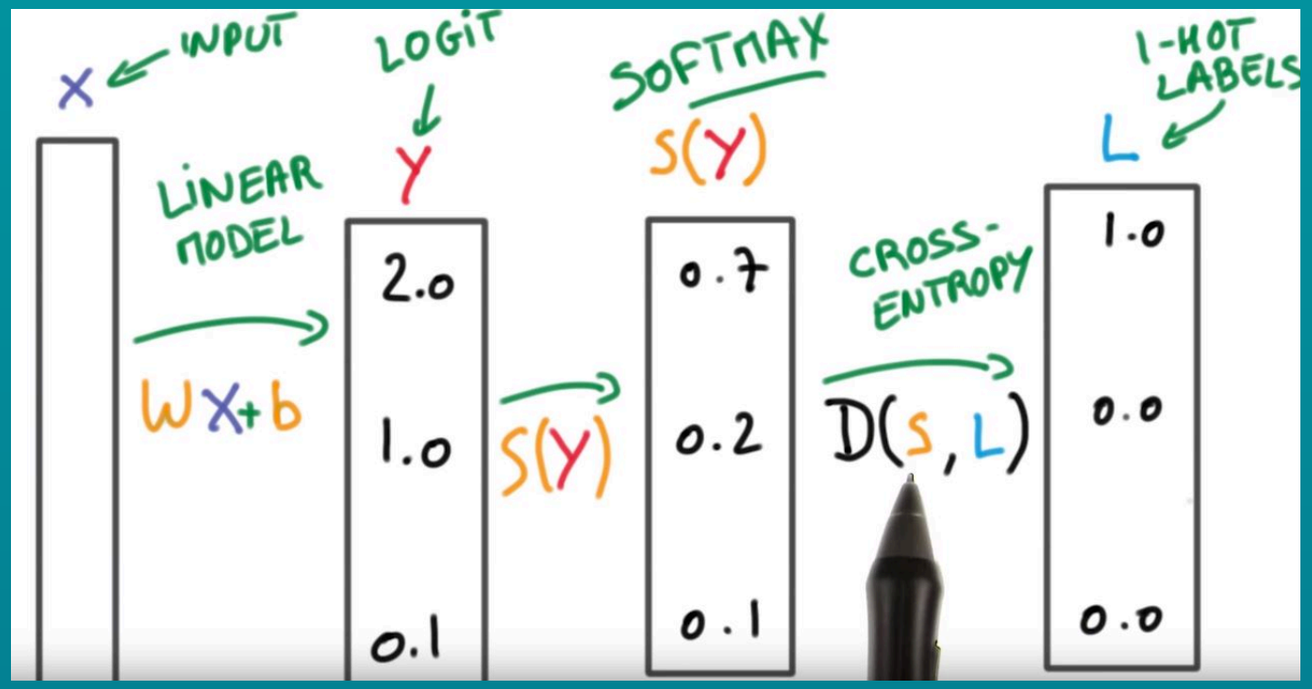


Image source: Google Tensorflow Course

Loss 函数之后?

- 利用backpropagation 进行训练
- 计算获得的梯度为n, 则相应的W更新为
  - W += -1 * learning_rate * W

示意图

Loss + 正则

- 为了使得模型中的参数大小比较均衡， 真正的Loss会加上所有参数的和：

- $$Loss = Loss + \lambda \sum_{p \in Parameters} W_p^2$$

如何训练

- 1. 从数据中选取部分数据， 利用模型进行拟合；
- 2. 计算出梯度， 进行反向传播；
- 3. 基于反向传播进行参数的更新;
- 4. 直到模型稳定

神经网络区别于其他机器学习模型的特点

- 1. 模型类别灵活；
- 2. 抽象能力强大；
- 3. 所需的数据量大；
- 4. 泛化能力强；

Assignment

- 1. 为什么数据在输入到模型之前要进行normalization 和 scaling?
- normaliaztion 使得标准差比较小的值（例如1）， 平均数为0；
- scaling 使得所有的值都在一定的范围内
- **调研这是为什么?**
- 2. 安装tensorflow 版本 >= 1.3
- Next: Tensorflow 的基本结构

Tensorflow 的基本知识

MiniTensorflow

- 1. 用纯的Python实现一个mini tensorflow

Assignment

- 1. 复现 mini-tensorflow 并将完成的版本 同步到自己的Github上；
- 2. 复现的时候， 竟可能的为代码添加注释。