lecture03 Word Window Classification,Neural Networks, and Matrix Calculus

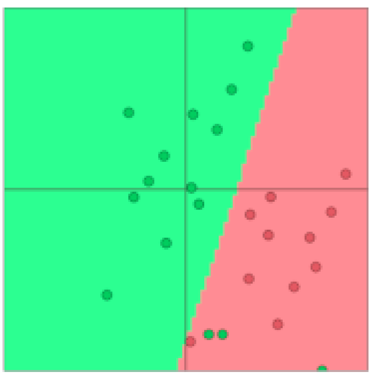
---

Classification setup and notation

一个包含如下样本的训练集：

$\{x_i, y_i\}_{i=1}^{N}$

xi：输入（词、向量、索引、句子、文档等），维度为d
yi：试图预测的标签（类别，共有C类）



Visualizations with ConvNetJS by Karpathy!
http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

the learned line is our classifier。
传统ML中：xi固定，训练softmax/logistic回归的权重

$W \in R^{C*d}$

来决定边界

- softmax classifier

$$p(y|x) = \frac{exp(W_y.x)}{\sum_{c=1}^{C} exp(W_c \cdot x)}$$

1. 将 W 的 y^{th} 行和 x 中的对应行相乘得到分数

$$W_y \cdot x = \sum_{i=1}^{d} W_{yi} x_i = f_y$$

计算所有的 $f_c, for\ c = 1, \ldots, C$

2. 使用softmax函数获得归一化的概率

$$p(y|x) = \frac{\exp(f_y)}{\sum_{c=1}^{C} \exp(f_c)} = \text{softmax}(f_y)$$

对于每个训练样本，目标是最大化正确类y的概率，或最小化负对数概率

$$-\log p(y|x) = -\log\left(\frac{\exp(f_y)}{\sum_{c=1}^{C} \exp(f_c)}\right)$$

Background: What is “cross entropy” loss/error?

- 交叉熵概念来源于信息论
- 真实概率分布p
- 计算的模型概率q
- 交叉熵为：

$$H(p, q) = -\sum_{c=1}^{C} p(c) log q(c)$$

- 假定目标的概率分布在正确的类上是1，在其他为0。p=[0,...,0,1,0,...,0]为独热向量，唯一剩下的项是真实类的负对数概率

Classification over a full dataset

整个数据集上的交叉熵损失函数

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} -log(\frac{e^{f_{y_i}}}{\sum_{c=1}^{C} e^{f_c}})$$

$$f = Wx$$

Traditional ML optimization

一般机器学习的学习参数由权重的列组成

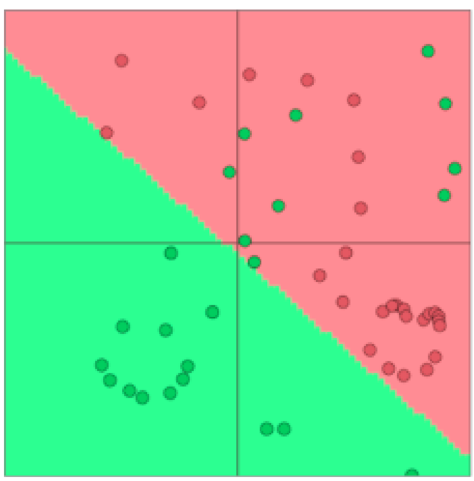
$$\theta = \begin{bmatrix} W_{\cdot 1} \\ \vdots \\ W_{\cdot d} \end{bmatrix} = W(:) \in \mathrm{R}^{Cd}$$

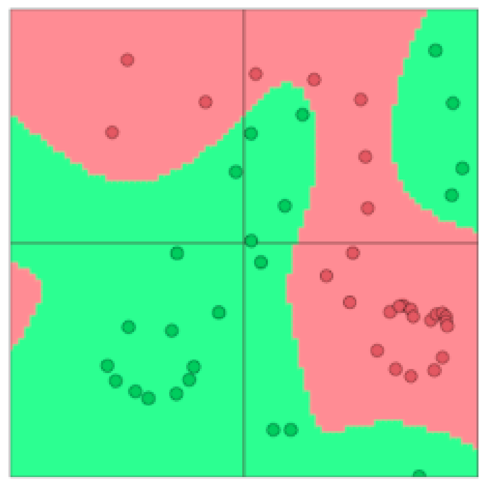
只通过以下方式更新决策边界

$$\nabla_\theta J(\theta) = \begin{bmatrix} \nabla W_{\cdot 1} \\ \vdots \\ \nabla W_{\cdot d} \end{bmatrix} = W(:) \in \mathrm{R}^{Cd}$$

Neural Network Classifiers

简单的线性分类器，并不强大，当问题复杂时是无用的
而神经网络可以学习更复杂的函数和非线性决策边界

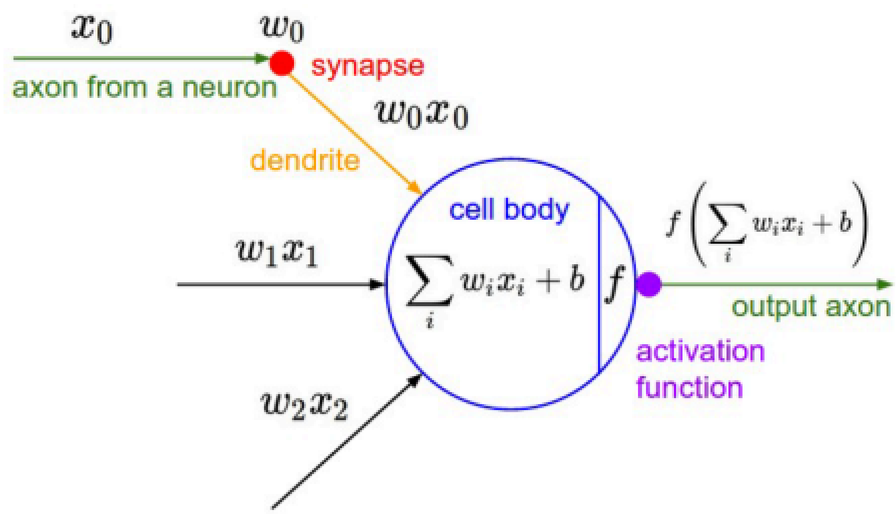




Classification difference with word vectors

DL中我们学习权重矩阵W和词向量x，传统参数和表示
需要对其进行更新优化分类器

An artificial neuron



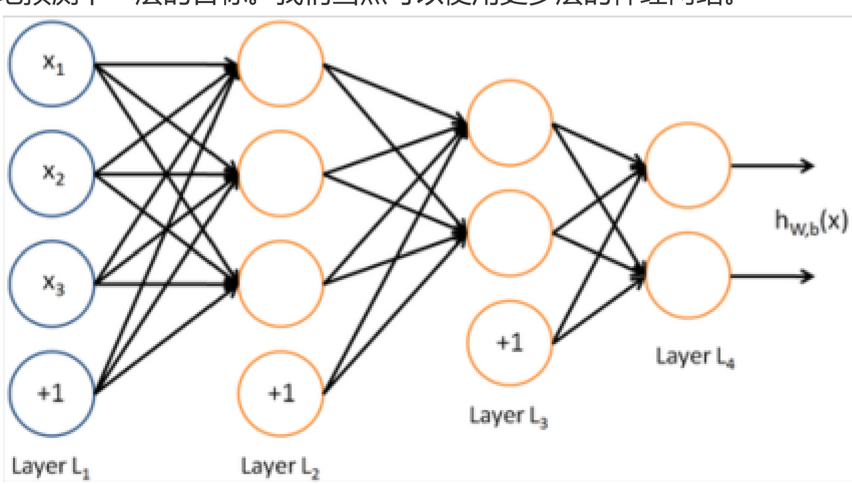
f = nonlinear activation fct. (e.g. sigmoid), w = weights, b = bias, h = hidden, x = inputs

$$h_{w,b}(x) = f(w^T x + b)$$

$$f(z) = \frac{1}{1 + e^{-z}}$$

A neural network= running several logistic regressions at the same time

如果我们输入一个向量通过一系列逻辑回归函数，那么我们得到一个输出向量，但是我们不需要提前决定这些逻辑回归试图预测的变量是什么。
我们可以输入另一个逻辑回归函数。损失函数将指导中间隐藏变量应该是什么，以便更好地预测下一层的目标。我们当然可以使用更多层的神经网络。



we have:

a1=f(W11x1+W12x2+W13x3+b1)

a2=f(W21x1+W22x2+W23x3+b2)

In matrix notation

z=Wx+b

a=f(z)激活函数

Activation f is applied element-wise:
  f([z1,z2,z3])=[f(z1),f(z2),f(z3)]
when we apply that to a vector, we apply it to each element of the vecotr element-wise.

Non-linearities (aka“f ”): Why they’re needed

- 没有非线性，深度神经网络只能做线性变换，且多个线性变换仅组成另一个简单线性变换
- 对于非线性函数，使用更多层可以近似更复杂的函数

Named Entity Recognition (NER)

任务：查找和分类文件中的实体名称
可能用途：

1. 跟踪文档中提到的特定的实体
2. 问题回答的答案常为命名实体
3. 很多需要的信息为命名实体间的联系
4. 其他分类

Why might NER be hard?

- 很难计算出实体边界
- 很难判断是否为实体
- 很难知道未知/新奇实体的类别
- 实体类别模棱两可，依赖于上下文

Binary word window classification

为在上下文中的语言构建分类器

Window classification

在相邻的上下文窗口对一个词进行分类，简单的方法是对窗口中的词向量取平均，并对平均向量进行分类，但是这会丢失位置信息

Softmax

例子: Not all museums in Paris are amazing .

$$X_{window} = [X_{museums} X_{in} X_{paris} X_{are} X_{amazing}]$$

将中心词上下文窗口大小至少为2的所有单词向量组成一个窗口列向量

- With $x = x_{window}$ we can use the same softmax classifier as before

predicted model
output probability

$$\boxed{\hat{y}_y} = p(y|x) = \frac{\exp(\boxed{W_y.x})}{\sum_{c=1}^{C}\exp(W_c.x)}$$

- With cross entropy error as before:

same

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} -\log\left(\frac{e^{\boxed{f_{y_i}}}}{\sum_{c=1}^{C} e^{f_c}}\right)$$

同样利用求导优化来更新向量
我们希望窗口为 [museums in Paris are amazing] 中心词为地点类别的（真正窗口）比窗口 [Not all museums in Paris] 不是地点类别的（损坏窗口）得分要高

Neural Network Feed-forward Computation

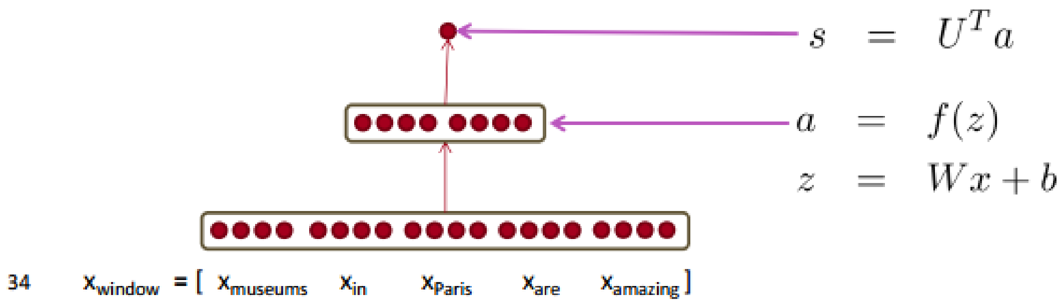
神经网络顶层a，与向量U点乘，返回即为得分

$$score(x) = U^T a \in R$$

下图为计算有三层神经网络的窗口得分

- $s = score$("museums in Paris are amazing")

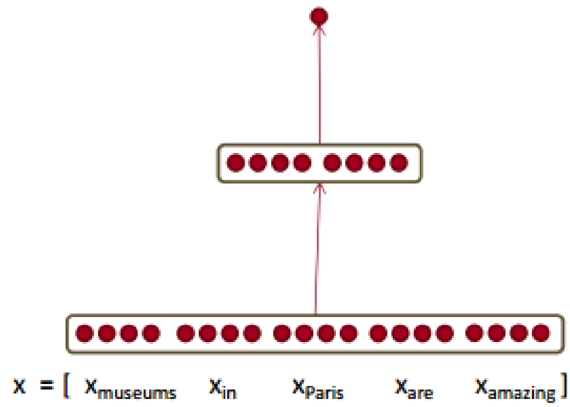
$$s = U^T f(Wx + b)$$
$$x \in \mathbb{R}^{20\times 1}, W \in \mathbb{R}^{8\times 20}, U \in \mathbb{R}^{8\times 1}$$



$$s = \boldsymbol{u}^T \boldsymbol{h}$$

$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

$$\boldsymbol{x} \quad \text{(input)}$$



Gradients

- 给定一个函数，有1个输出和 n 个输入
  $f(x)=f(x_1,x_2,\ldots,x_n)$
  梯度是关于每个输入的偏导数的向量
  $$\frac{\partial f}{\partial x} = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}]$$
- 给定一个函数，有 m 个输出和 n 个输入
  $f(x)=[f_1(x_1,x_2,\ldots,x_n),\ldots,f_m(x_1,x_2,\ldots,x_n)]$
  其雅可比矩阵是一个m×n的偏导矩阵
  $$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$\left(\frac{\partial f}{\partial x}\right)_{ij} = \frac{\partial f_i}{\partial x_j}$$

$$\boldsymbol{h} = f(\boldsymbol{z}), \text{what is } \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}? \qquad \boldsymbol{h}, \boldsymbol{z} \in \mathbb{R}^n$$
$$h_i = f(z_i)$$

$$\boldsymbol{h} = f(\boldsymbol{z}), \text{what is } \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}? \qquad \boldsymbol{h}, \boldsymbol{z} \in \mathbb{R}^n$$
$$h_i = f(z_i)$$

$$\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}\right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i) \qquad \text{definition of Jacobian}$$

$$\boldsymbol{h} = f(\boldsymbol{z}), \text{what is } \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}? \qquad \boldsymbol{h}, \boldsymbol{z} \in \mathbb{R}^n$$
$$h_i = f(z_i)$$

$$\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}\right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i) \qquad \text{definition of Jacobian}$$
$$= \begin{cases} f'(z_i) & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases} \qquad \text{regular 1-variable derivative}$$

$$\boldsymbol{h} = f(\boldsymbol{z}), \text{what is } \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}? \qquad \boldsymbol{h}, \boldsymbol{z} \in \mathbb{R}^n$$
$$h_i = f(z_i)$$

$$\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}\right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i) \qquad \text{definition of Jacobian}$$
$$= \begin{cases} f'(z_i) & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases} \qquad \text{regular 1-variable derivative}$$

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} = \begin{pmatrix} f'(z_1) & & 0 \\ & \ddots & \\ 0 & & f'(z_n) \end{pmatrix} = \text{diag}(\boldsymbol{f}'(\boldsymbol{z}))$$

49

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{W}$$
$$\frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{I} \quad \text{(Identity matrix)}$$
$$\frac{\partial}{\partial \boldsymbol{u}}(\boldsymbol{u}^T \boldsymbol{h}) = \boldsymbol{h}^{\boldsymbol{T}}$$

$$s = \boldsymbol{u}^T \boldsymbol{h}$$
$$\boldsymbol{h} = f(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{x} \quad \text{(input)}$$

$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$
$$= \boldsymbol{u}^{\boldsymbol{T}} \text{diag}(f'(\boldsymbol{z}))\boldsymbol{I}$$
$$= \boldsymbol{u}^{\boldsymbol{T}} \circ f'(\boldsymbol{z})$$

Useful Jacobians from previous slide

$$\frac{\partial}{\partial \boldsymbol{u}}(\boldsymbol{u}^T \boldsymbol{h}) = \boldsymbol{h}^T$$
$$\frac{\partial}{\partial \boldsymbol{z}}(f(\boldsymbol{z})) = \text{diag}(f'(\boldsymbol{z}))$$
$$\frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{I}$$

65

同理，当计算s对W的导数时：

$$\frac{\partial s}{\partial \boldsymbol{W}} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}}$$

$$\frac{\partial s}{\partial \boldsymbol{b}} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$

$$\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}}$$

$$\frac{\partial s}{\partial \boldsymbol{b}} = \boldsymbol{\delta} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}} = \boldsymbol{\delta}$$

$$\boldsymbol{\delta} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} = \boldsymbol{u}^T \circ f'(\boldsymbol{z})$$

δ is local error signal

δ 是局部误差符号

∂s/∂W的形状是 n×m

- Remember $\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}}$
  - $\boldsymbol{\delta}$ is going to be in our answer
  - The other term should be $\boldsymbol{x}$ because $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$
- It turns out $\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta}^T \boldsymbol{x}^T$

δ is local error signal at z
x is local input signal

Why the Transposes?

$$\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta}^T \boldsymbol{x}^T = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_n \end{bmatrix} [x_1, ..., x_m] = \begin{bmatrix} \delta_1 x_1 & \dots & \delta_1 x_m \\ \vdots & \ddots & \vdots \\ \delta_n x_1 & \dots & \delta_n x_m \end{bmatrix}$$

$$\underset{[n \times m]}{\frac{\partial s}{\partial \boldsymbol{W}}} = \underset{[n \times 1]}{\boldsymbol{\delta}^T} \underset{[1 \times m]}{\boldsymbol{x}^T}$$

详见note