Word Window Classification

交叉熵损失函数

Background: What is "cross entropy" loss/error?

- Concept of "cross entropy" is from information theory
- Let the true probability distribution be *p*
- Let our computed model probability be *q*
- The cross entropy is:

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

- Assuming a ground truth (or true or gold or target) probability distribution that is 1 at the right class and 0 everywhere else: *p* = [0,...,0,1,0,...0] then:
- **Because of one-hot *p*, the only term left is the negative log probability of the true class**

11

上图是交叉熵的公式，在逻辑回归中，$p(c)$就是y和$1-y$, $q(c)$就是$h(x)$和$1-h(x)$。

Classification over a full dataset

- Cross entropy loss function over full dataset $\{x_i, y_i\}_{i=1}^N$

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

- Instead of

$$f_y = f_y(x) = W_y.x = \sum_{j=1}^{d} W_{yj}x_j$$

We will write f in matrix notation:

$$f = Wx$$

上图是真实training时的损失函数，实际情况中$p(c)$为1。

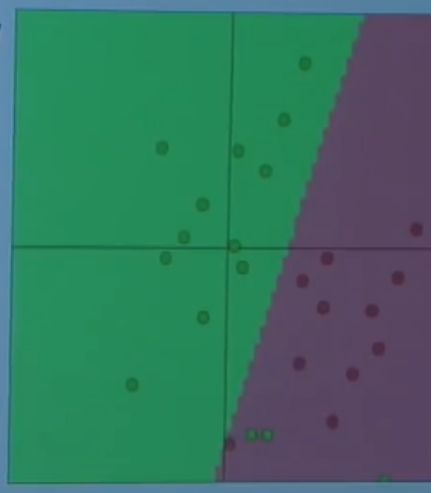
Traditional ML optimization

- For general machine learning θ usually only consists of columns of W:

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

- So we only update the decision boundary via

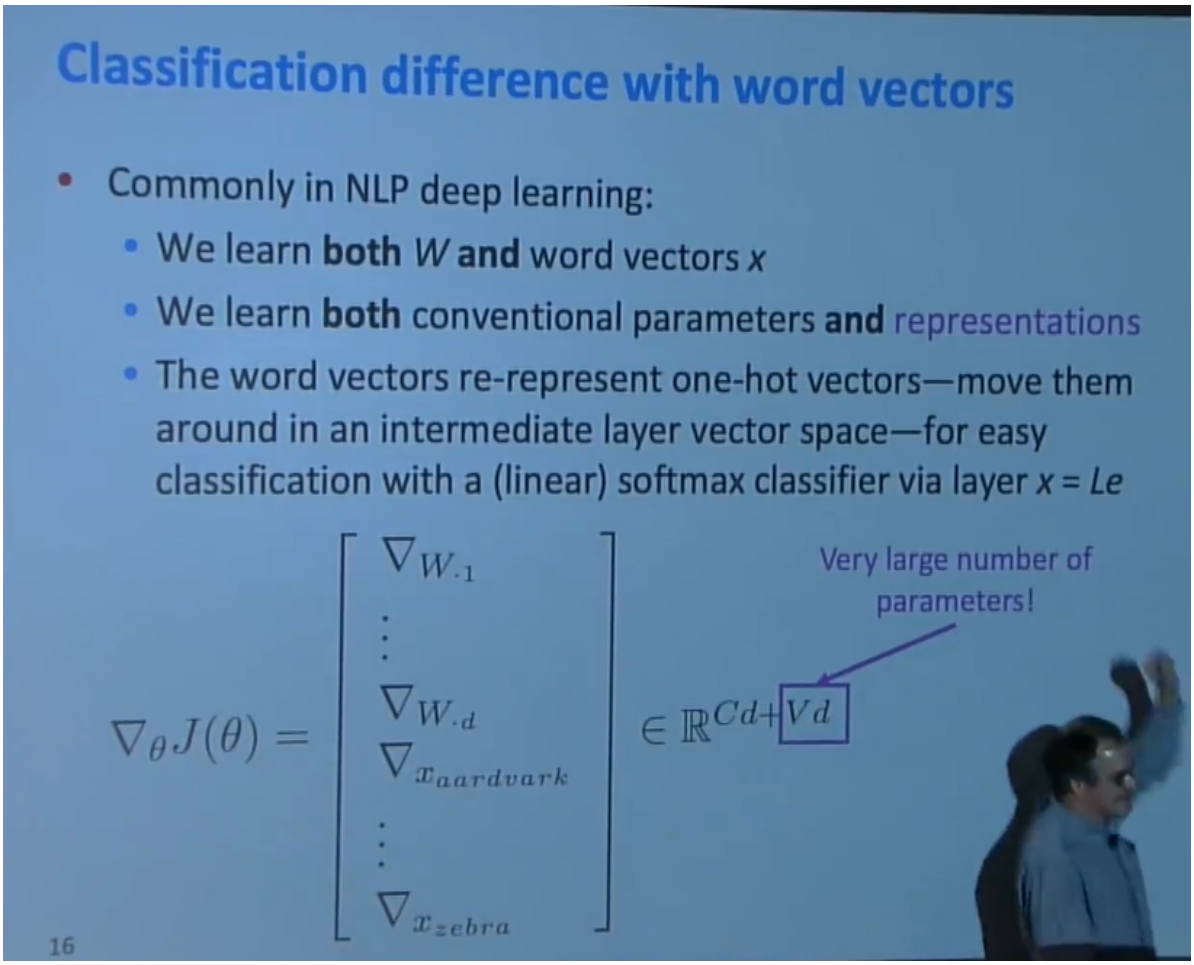
$$\nabla_\theta J(\theta) = \begin{bmatrix} \nabla_{W_{\cdot 1}} \\ \vdots \\ \nabla_{W_{\cdot d}} \end{bmatrix} \in \mathbb{R}^{Cd}$$



Visualizations with ConvNetJS by Karpathy

上图是隐藏层的参数矩阵$W \in R^{Cd}$，即输入$X \in R^{md}$，乘以W，得到$O \in R^{mC}$。

Classification difference with word vectors

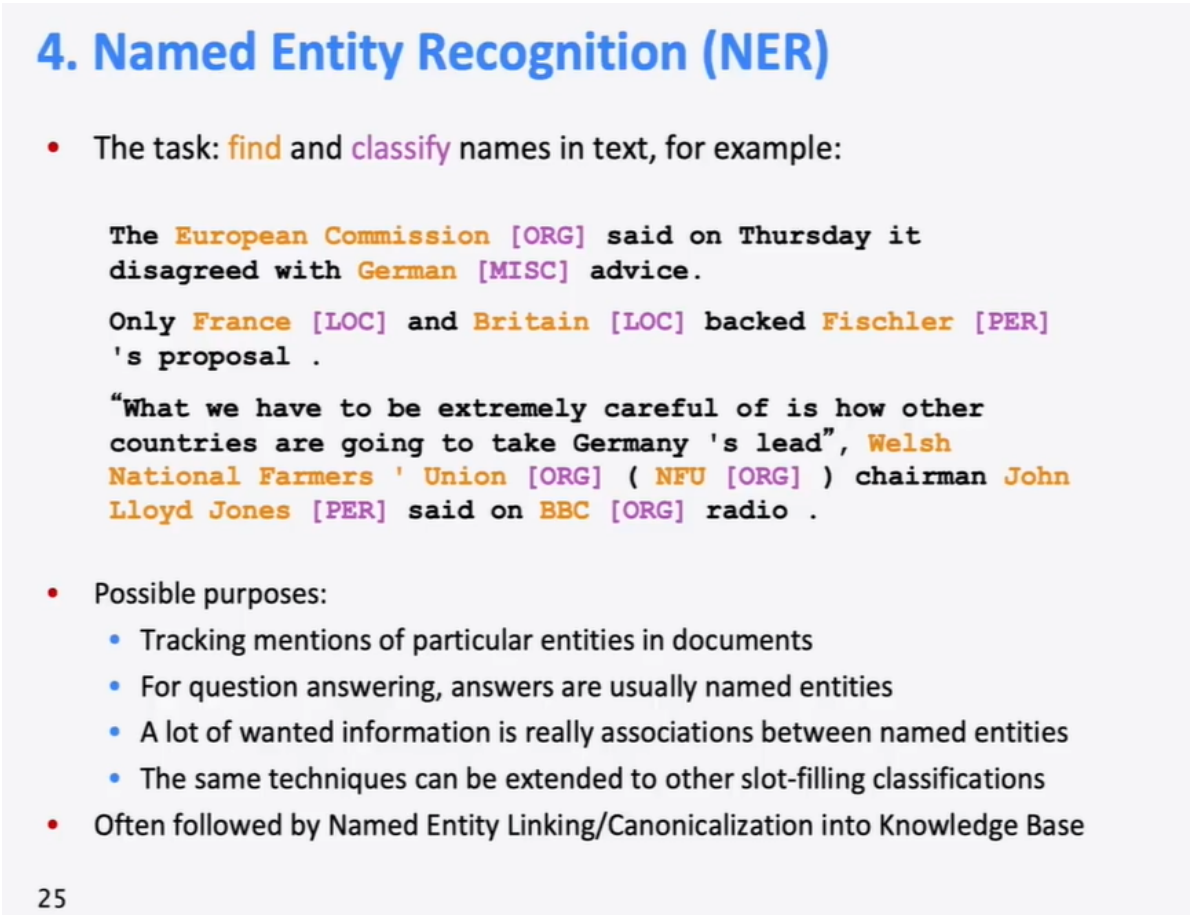


上图是用NN分类的区别: 不仅要学习W，还要同时学习词向量x。

NN的简介

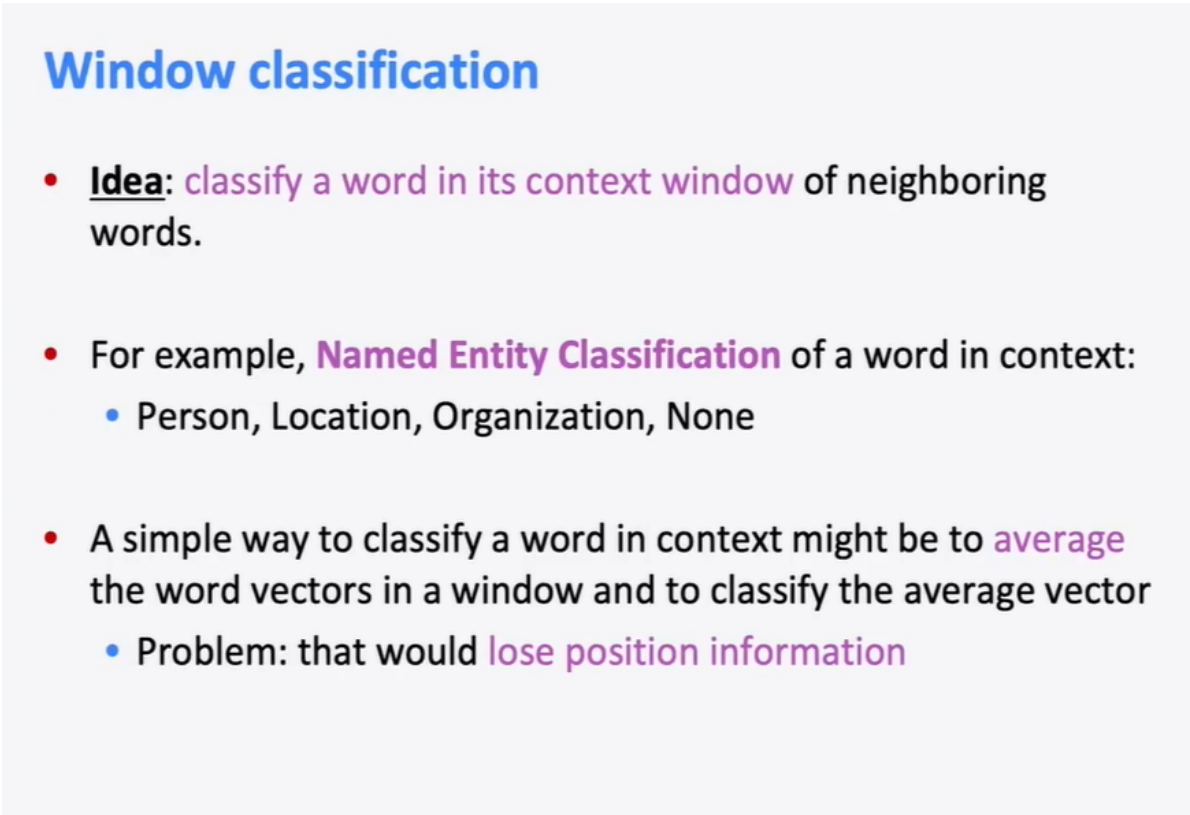
略

Named Entity Recognition (NER)



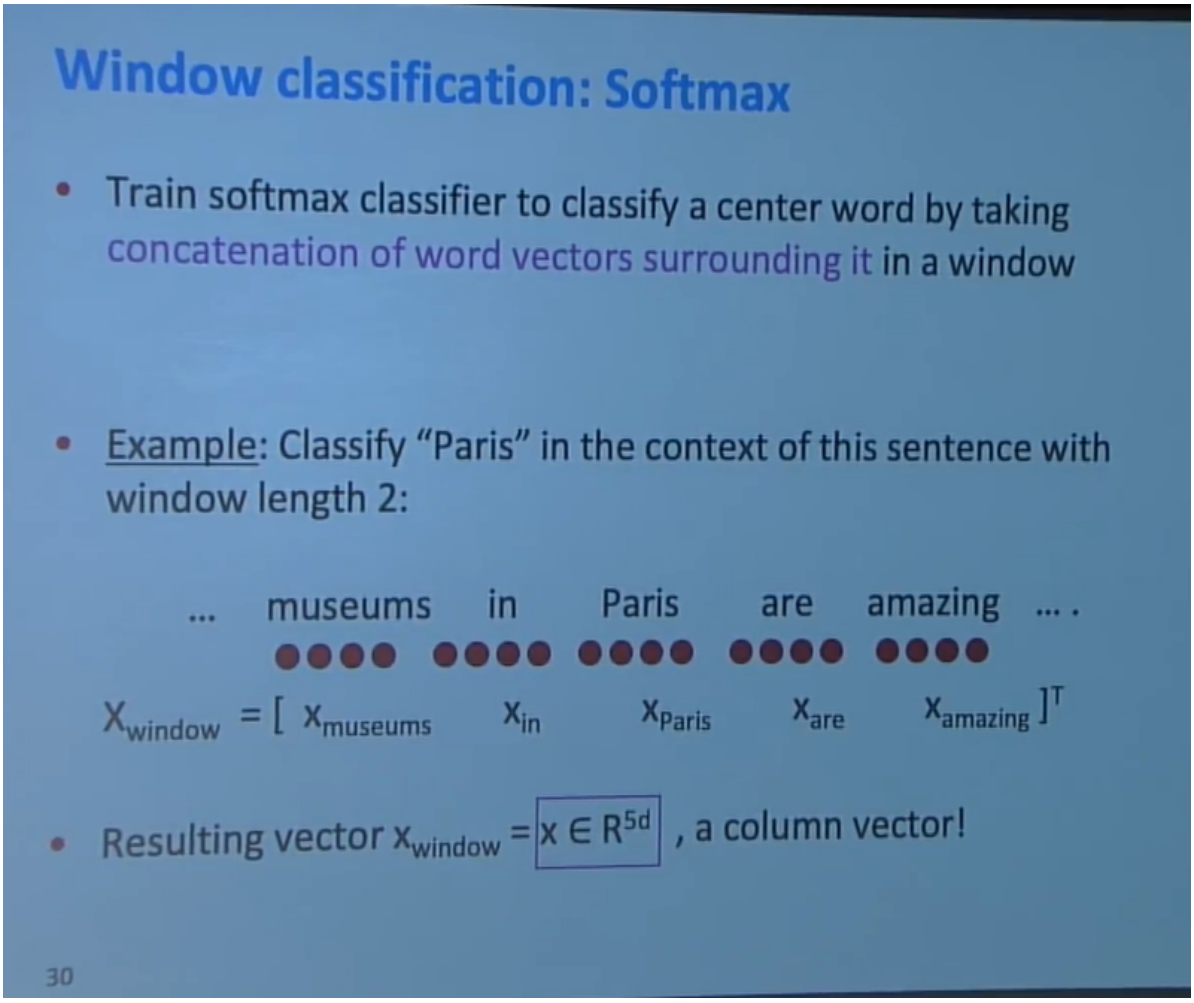
Binary word window classification

一般不会对single word分类，而是放在context中进行分类。



如果直接将一个window取平均，会损失位置信息。

一种改进方法是把window的wordvec都concat起来:



这是一篇经典论文，ICML2018的test of time奖。

Matrix calculus review

多元求导

分母标量，分子向量

Gradients

- Given a function with 1 output and *n* inputs

$$f(\boldsymbol{x}) = f(x_1, x_2, ..., x_n)$$

- It's gradient is a vector of partial derivatives with respect to each input

$$\frac{\partial f}{\partial \boldsymbol{x}} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}\right]$$

42

分母m维向量，分子n维向量 (m个输出，n个输入)

Jacobian Matrix: Generalization of the Gradient

- Given a function with ***m* outputs** and *n* inputs

$$\boldsymbol{f}(\boldsymbol{x}) = [f_1(x_1, x_2, ..., x_n), ..., f_m(x_1, x_2, ..., x_n)]$$

- It's Jacobian is an ***m* x *n* matrix** of partial derivatives

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

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

43

分母标量，分子矩阵 (遵循shape convention)

Derivative with respect to Matrix

- What does $\frac{\partial s}{\partial \boldsymbol{W}}$ look like? $\boldsymbol{W} \in \mathbb{R}^{n \times m}$
- 1 output, *nm* inputs: 1 by *nm* Jacobian?
  - Inconvenient to do $\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$
- Instead follow convention: shape of the gradient is shape of parameters
  - So $\frac{\partial s}{\partial \boldsymbol{W}}$ is *n* by *m*: $\begin{bmatrix} \frac{\partial s}{\partial W_{11}} & \cdots & \frac{\partial s}{\partial W_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial s}{\partial W_{n1}} & \cdots & \frac{\partial s}{\partial W_{nm}} \end{bmatrix}$

70

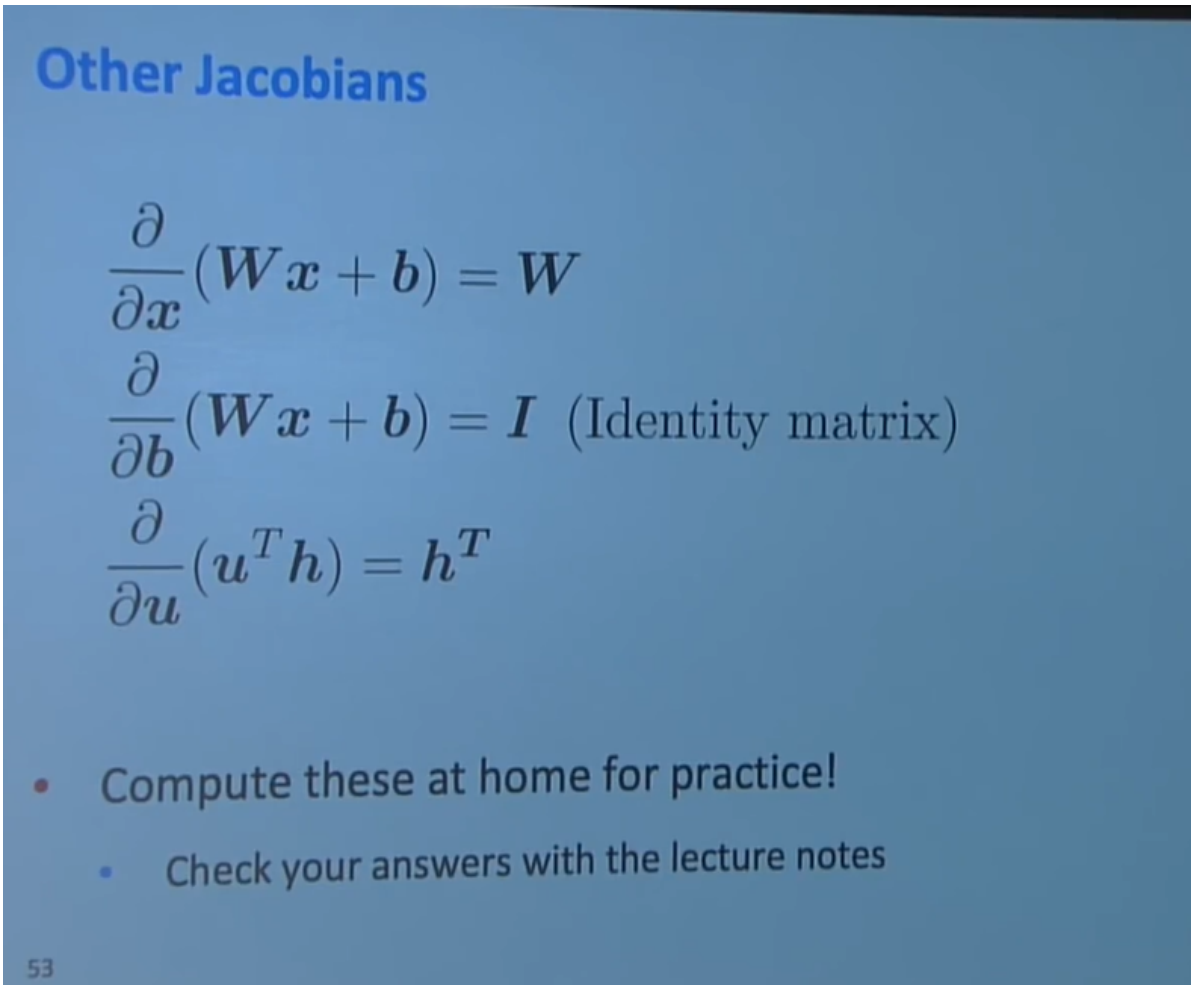
Derivative with respect to Matrix

- 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 $\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta}^T \boldsymbol{x}^T$

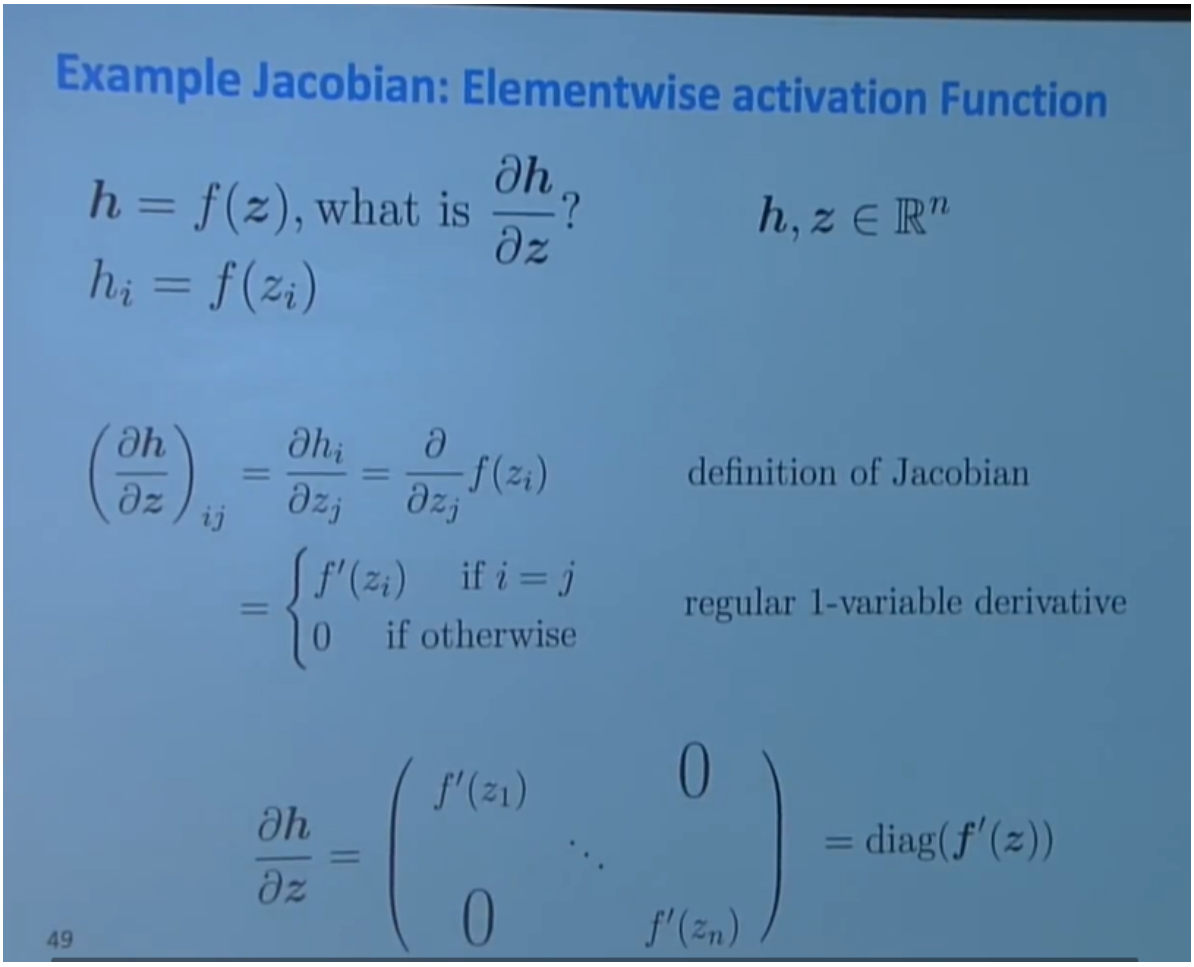
δ is local error signal at z
x is local input signal

71

其他


Other Jacobians
$\frac{\partial}{\partial x}(Wx + b) = W$
$\frac{\partial}{\partial b}(Wx + b) = I$ (Identity matrix)
$\frac{\partial}{\partial u}(u^T h) = h^T$
Compute these at home for practice!
Check your answers with the lecture notes
53

一个例子



上图是激活函数，所以$h_i = f(z_i)$，不用考虑其他z_j，也因此最后结果是对角矩阵。

重复计算部分可以保留

Re-using Computation

- Suppose we now want to compute $\frac{\partial s}{\partial W}$
  - Using the chain rule again:

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

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

The same! Let's avoid duplicated computation...

67

Shape convention

What shape should derivatives be?

- $\frac{\partial s}{\partial b} = h^T \circ f'(z)$ is a row vector
  - But convention says our gradient should be a column vector because b is a column vector...
- Disagreement between Jacobian form (which makes the chain rule easy) and the shape convention (which makes implementing SGD easy)
  - We expect answers to follow the **shape convention**
  - But Jacobian form is useful for computing the answers

74