Neural Networks III

CMSC 422

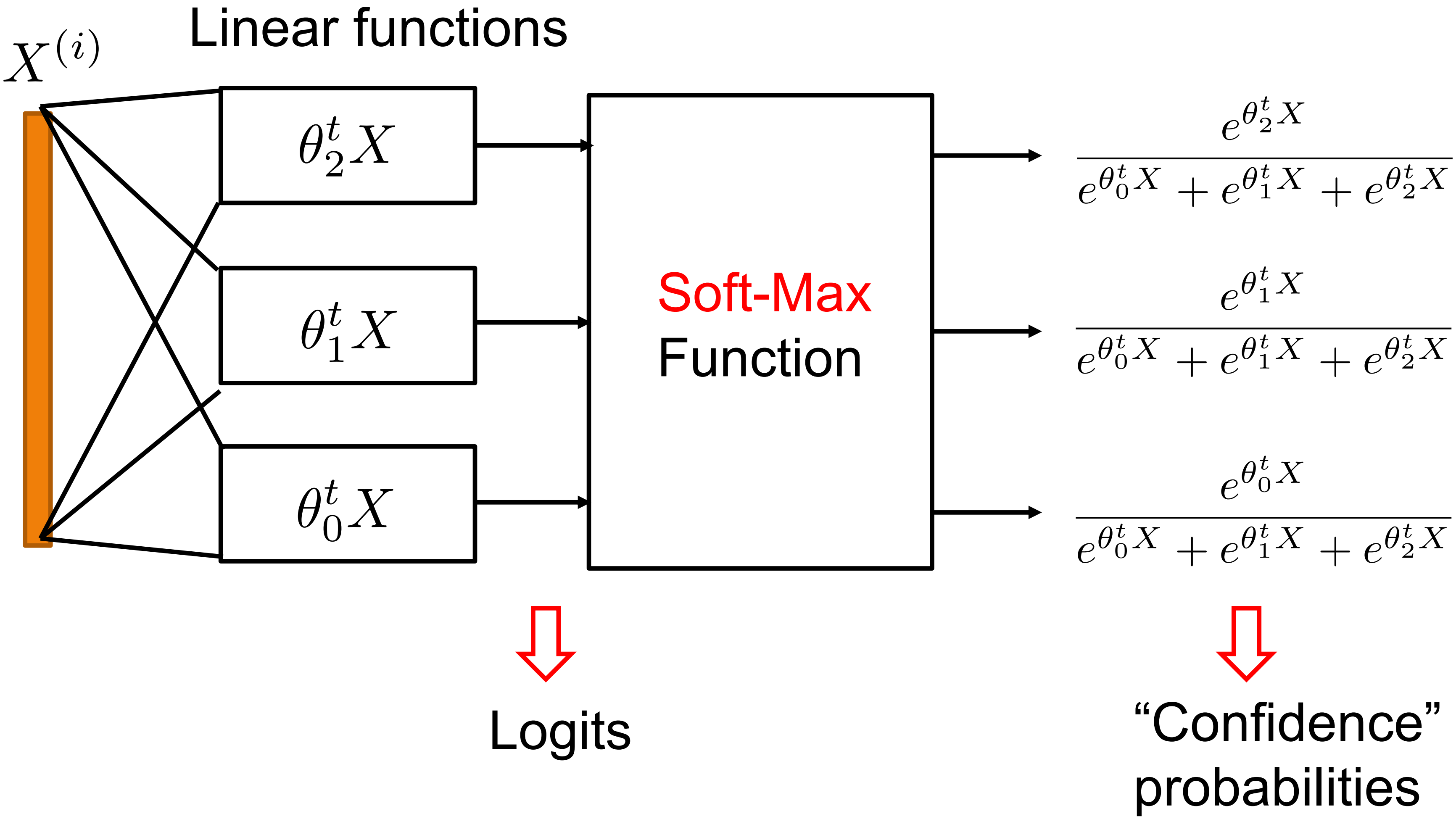
SOHEIL FEIZI

sfeizi@cs.umd.edu

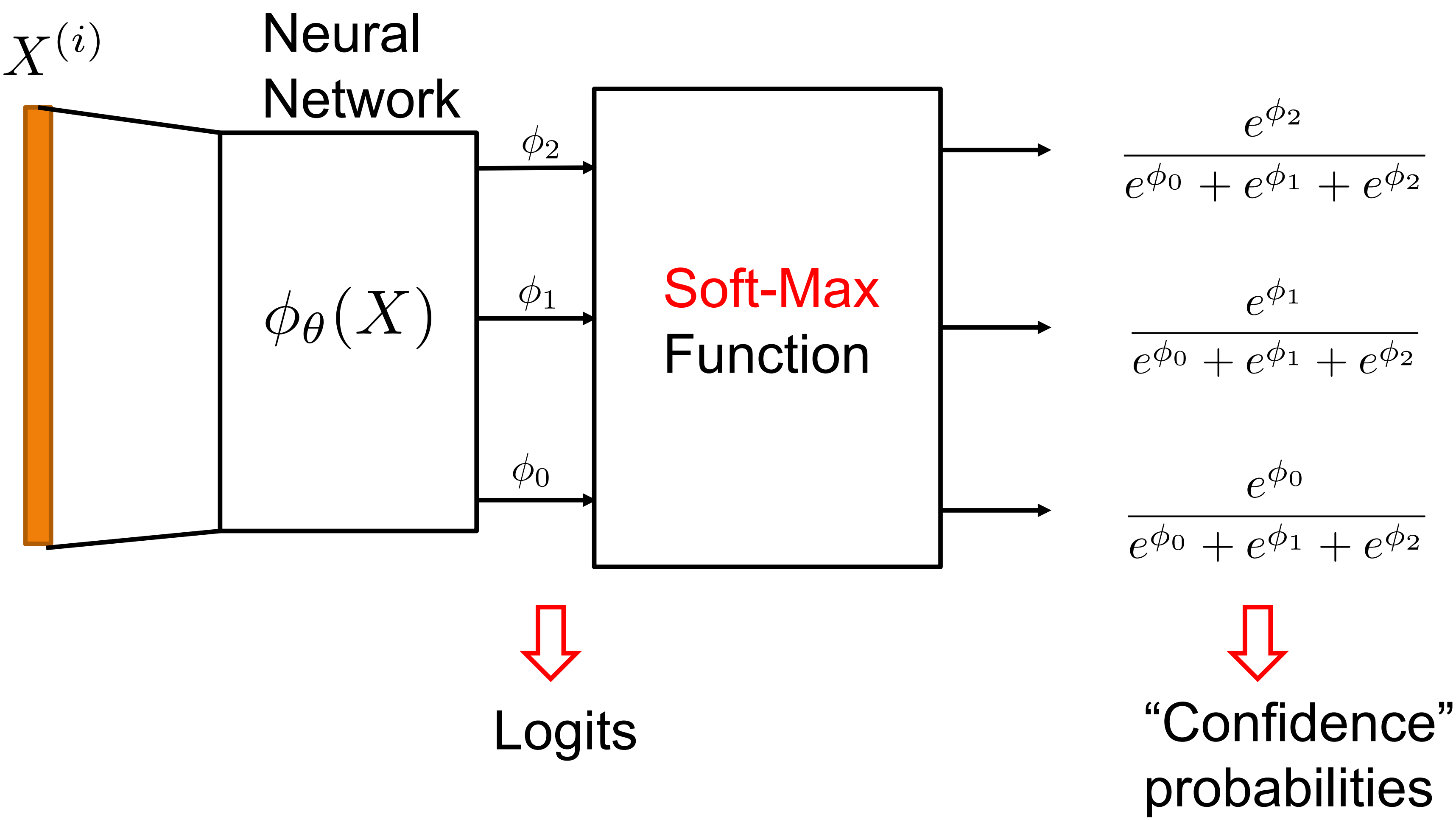
Multi-Label Classification

Q: how to extend our method for multi-label classification?

Recall: Multi-Label Classification using Logistic Regression

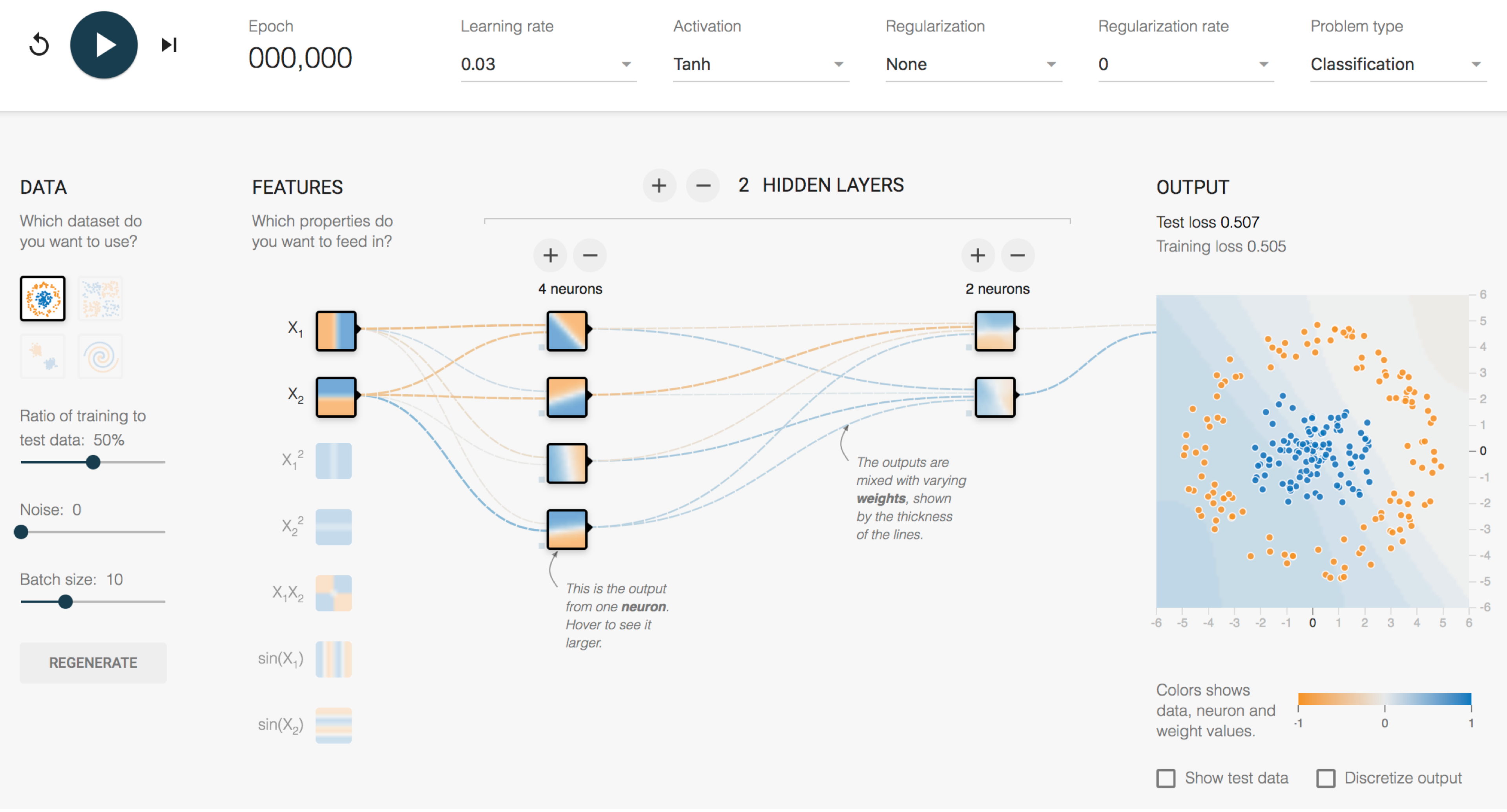


Multi-Label Classification Using NNs



Try different architectures and training parameters here:

http://playground.tensorflow.org


Epoch
000,000
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X1^2
X2^2
X1X2
sin(X1)
sin(X2)
2 HIDDEN LAYERS
4 neurons
2 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.507
Training loss 0.505
Colors shows data, neuron and weight values.
Show test data
Discretize output

Tricky issues with neural network training

- Sensitive to initialization
  - Objective is non-convex, many local optima
  - In practice: start with random values rather than zeros
- Many other hyper-parameters
  - Number of hidden units (and potentially hidden layers)
  - Gradient descent learning rate
  - Stopping criterion

Neural networks vs. linear classifiers

Advantages of Neural Networks:

– More expressive

– Less feature engineering

Challenges using Neural Networks:

– Harder to train

– Harder to interpret

Neural Network Architectures

- We focused on a **multi-layer feedforward** network
- Many other deeper architectures
  - Convolutional networks
  - Recurrent networks (LSTMs)
  - Dense Nets, ResNets, etc

Issues in Deep Neural Networks

- Long training time
  - There are sometimes a lot of training data
  - Many iterations (epochs) are typically required for optimization
  - Computing gradients in each iteration takes too much time

Slide credit: adapted from Bohyung Han

Improving on Gradient Descent: Stochastic Gradient Descent (SGD)

- Update weights for each example

$$E = \frac{1}{2}(y^n - \hat{y}^n)^2 \qquad \boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) - \epsilon \frac{\partial E^n}{\partial \boldsymbol{w}_i}$$

+ **Fast, online**
− **Sensitive to noise**

- Mini-batch SGD: Update weights for a small set of examples

$$E = \frac{1}{2}\sum_{n \in B}(y^n - \hat{y}^n)^2 \qquad \boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) - \epsilon \frac{\partial E^B}{\partial \boldsymbol{w}_i}$$

+ **Fast, online**
+ **Robust to noise**

Slide credit: Bohyung Han

Improving on Gradient Descent: SGD with Momentum

- Update based on gradients + previous direction

$$v_i(t) = \alpha v_i(t-1) - (1-\alpha)\frac{\partial E}{\partial w_i}(t)$$

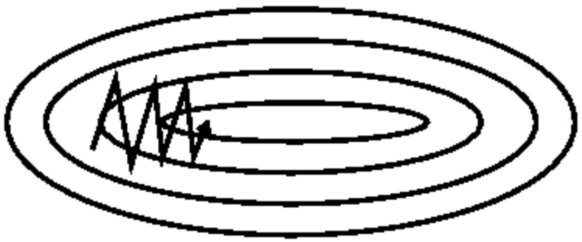
$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \boldsymbol{v}(t)$$

+ Converge faster
+ Avoid oscillation

Slide credit: Bohyung Han

Improving on Gradient Descent: SGD with Momentum

SGD w/o momentum



SGD with momentum helps dampen oscillations

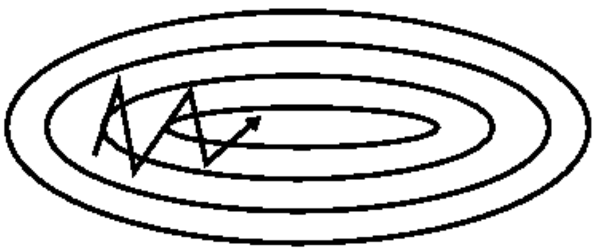


Image: http://ruder.io/optimizing-gradient-descent/index.html#momentum

Improving the Training Objective: Regularization/Weight Decay

- Penalize the size of the weights

$$C = E + \frac{\lambda}{2}\sum_i w_i^2$$

$$w_i(t+1) = w_i(t) - \epsilon\frac{\partial C}{\partial w_i} = w_i(t) - \epsilon\frac{\partial E}{\partial w_i} - \lambda w_i$$

→ **Improves generalization**

Slide credit: Bohyung Han

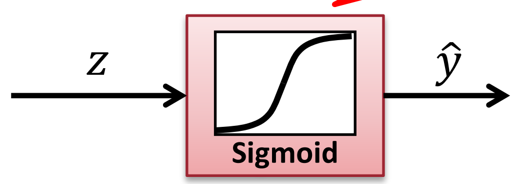
Vanishing Gradient Problem

In deep networks

– Gradients in the lower layers are typically extremely small

– Optimizing multi-layer neural networks takes huge amount of time

Derivative of sigmoid in [0,1]



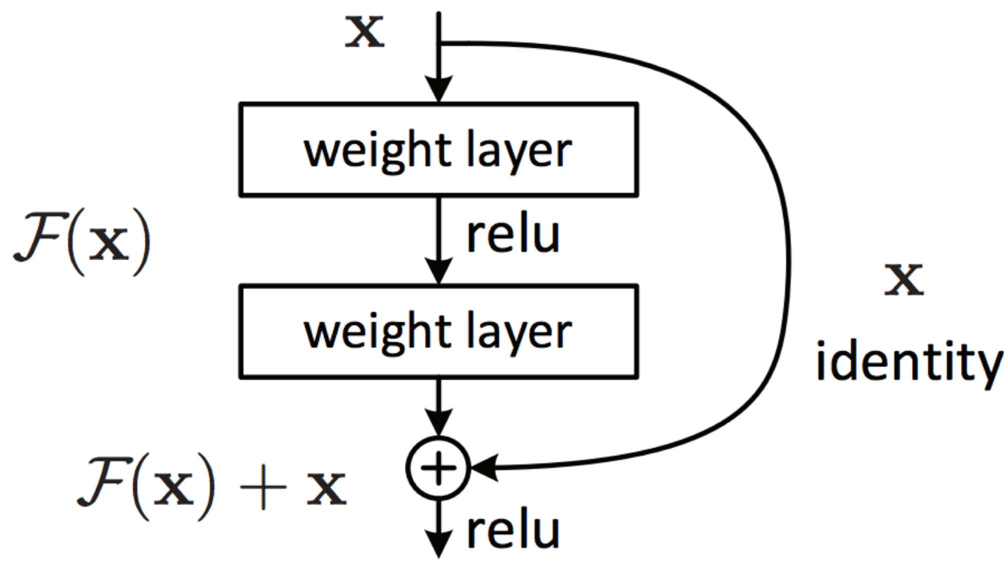
$$\frac{\partial E}{\partial w_{ki}} = \sum_n \frac{\partial z_i^n}{\partial w_{ki}} \frac{d\hat{y}_i^n}{dz_i^n} \frac{\partial E}{\partial \hat{y}_i^n} = \sum_n \frac{\partial z_i^n}{\partial w_{ki}} \boxed{\frac{d\hat{y}_i^n}{dz_i^n}} \sum_j w_{ij} \boxed{\frac{d\hat{y}_j^n}{dz_j^n}} \frac{\partial E}{\partial \hat{y}_j^n}$$

Slide credit: adapted from Bohyung Han

Vanishing Gradient Problem

- Vanishing gradient problem can be mitigated
  - Using custom neural network architectures
  - Using other non-linearities
    - E.g., Rectifier: $f(x) = \max(0,x)$

ResNet



Deep Residual Learning for Image Recognition

https://arxiv.org › cs

by K He - 2015 - Cited by 19999 - Related articles

Dec 10, 2015 - Abstract: Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are ...