Neural Networks and Autodifferentiation

CMSC 678
UMBC
February 14th, 2018
Recap from last time...
Experimental Design & Evaluation

DEVELOP ON DEV DATA

DON'T ITERATE ON YOUR TEST DATA
Classification Evaluation: Accuracy, Precision, and Recall

**Accuracy**: % of items correct
\[
\frac{TP + TN}{TP + FP + FN + TN}
\]

**Precision**: % of selected items that are correct
\[
\frac{TP}{TP + FP}
\]

**Recall**: % of correct items that are selected
\[
\frac{TP}{TP + FN}
\]

<table>
<thead>
<tr>
<th>Actually Correct</th>
<th>Actually Incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Selected/Guessed</strong></td>
<td>True Positive (TP)</td>
</tr>
<tr>
<td><strong>Not select/not guessed</strong></td>
<td>False Negative (FN)</td>
</tr>
</tbody>
</table>
Maximum Entropy (Log-linear) Models

\[ p(x | y) \propto \exp(\theta^T f(x, y)) \]

“model the posterior probabilities of the K classes via linear functions in \( \theta \), while at the same time ensuring that they sum to one and remain in [0, 1]” ~ Ch 4.4

“[The log-linear estimate] is the least biased estimate possible on the given information; i.e., it is maximally noncommittal with regard to missing information.” Jaynes, 1957
Normalization for Classification

\[ Z = \sum \exp \left( \begin{array}{c}
\text{label } X \\
\text{weight}_1 \cdot f_1(\text{fatally shot, } X) \\
\text{weight}_2 \cdot f_2(\text{seriously wounded, } X) \\
\text{weight}_3 \cdot f_3(\text{Shining Path, } X) \\
\text{...}
\end{array} \right) \]
Connections to Other Techniques

Log-Linear Models

(Multinomial) logistic regression
Softmax regression
Max`imum Entropy models (MaxEnt)
Generalized Linear Models
Discriminative Naïve Bayes
Very shallow (sigmoidal) neural nets

\[ y = \sum_{k} \theta_k x_k + b \]

the response can be a general (transformed) version of another response

logistic regression

\[ \frac{\log p(x = i)}{\log p(x = K)} = \sum_{k} \theta_k f(x_k, i) + b \]
Log-Likelihood Gradient

Each component $k$ is the difference between:

- the total value of feature $f_k$ in the training data

  $$\sum_i f_k(x_i, y_i)$$

- and

- the total value the current model $p_\theta$ *thinks* it computes for feature $f_k$

  $$\sum_i \mathbb{E}_p[f(x', y_i)]$$
Outline

Neural networks: non-linear classifiers

Learning weights: backpropagation of error

Autodifferentiation (in reverse mode)
The Sigmoid function is defined as:

$$\sigma(v) = \frac{1}{1 + \exp(-sv)}$$
\[ \sigma(v) = \frac{1}{1 + \exp(-sv)} \]

\[ \frac{\partial \sigma(v)}{\partial v} = s \cdot \sigma(v) \cdot (1 - \sigma(v)) \]

calc practice: verify for yourself
Linear Regression/Perceptron

\[ y = w^T x + b \]

output:
if \( y > 0 \): class 1
else: class 2
Linear Regression/Perceptron: A Per-Class View

\[ y = \mathbf{w}_1^T \mathbf{x} + b \]
\[ y = \mathbf{w}_2^T \mathbf{x} + b \]

output:
- if \( y > 0 \): class 1
- else: class 2

output:
- \( i = \text{argmax}\{y_1, y_2\} \)
- class \( i \)

binary version is special case
Logistic Regression/Classification

\[ y = \sigma(w^Tx + b) \]

\[ y = \text{softmax}(w^Tx + b) \]

\[ y_1 \propto \exp(w_1^Tx + b) \]

\[ y_2 \propto \exp(w_2^Tx + b) \]

output:

\[ i = \text{argmax} \{y_1, y_2\} \]

class \( i \)
Logistic Regression/Classification

**Q:** Why didn’t our maxent formulation from last class have multiple weight vectors?

**Output:**

\[ i = \text{argmax} \{ y_1, y_2 \} \]

**Class** \( i \)
Logistic Regression/Classification

Q: Why didn’t our maxent formulation from last class have multiple weight vectors?

A: Implicitly it did. Our formulation was
\[ y \propto \exp(w^T f(x, y)) \]

output:
\[ i = \arg\max \{y_1, y_2\} \]
class \( i \)
Stacking Logistic Regression

Goal: you still want to predict $y$

Idea: Can making an initial round of separate (independent) binary predictions $h$ help?

$h_i = \sigma(w_i^T x + b_0)$
Stacking Logistic Regression

\[ y_j = \text{softmax}(\beta_j^T h + b_1) \]

Predict \( y \) from your first round of predictions \( h \)

\[ h_i = \sigma(w_i^T x + b_0) \]

Idea: data/signal compression
Stacking Logistic Regression

\[ h_i = \sigma(w_i^T x + b_0) \quad \quad \quad y_j = \text{softmax}(\beta_j^T h + b_1) \]

Do we need (binary) probabilities here?
Stacking Logistic Regression

\[ h_i = F(w_i^T x + b_0) \]
\[ y_j = \text{softmax}(\beta_j^T h + b_1) \]

- \( F \): (non-linear) activation function
- Do we need probabilities here?
Stacking Logistic Regression

\[ h_i = F(w_i^T x + b_0) \]

\[ h \]

\[ y_j = \text{softmax}(\beta_j^T h + b_1) \]

Do we need probabilities here?

Classification: probably
Regression: not really

\( F \): (non-linear) activation function
Stacking Logistic Regression

\[ h_i = F(w_i^T x + b_0) \]
\[ y_j = G(\beta_j^T h + b_1) \]

- **F**: (non-linear) activation function
- **G**: (non-linear) activation function
- Classification: softmax
- Regression: identity
Feed-Forward Neural Network:
Multilayer Perceptron

\[ h_i = F(w_i^T x + b_0) \]

\[ y_j = G(\beta_j^T h + b_1) \]

\( F \): (non-linear) activation function

\( G \): (non-linear) activation function

Classification: softmax
Regression: identity
Feed-Forward Neural Network

\[ h_i = F(w_i^T x + b_0) \]

\[ y_j = G(\beta_j^T h + b_1) \]

\( w \): # hidden X # input

\( \beta \): # output X # hidden
Why Non-Linear?

\[ y_j = G(\beta_j^T h + b_1) \]

\[ y_j = G \left( \beta_j^T \left( F(w_i^T x + b_0) \right)_i \right) \]
Feed-Forward

$x$ \rightarrow \mathbf{h} \rightarrow \mathbf{y}$

\[ y_1, y_2 \]

$\mathbf{x}$
$\mathbf{h}$
$\mathbf{y}$

$\beta$

Information/computation flow

No self-loops (recurrence/reuse of weights)
Why “Neural?”

argue from neuroscience perspective

neurons (in the brain) receive input and “fire” when sufficiently excited/activated
Universal Function Approximator

**Theorem** [Kurt Hornik et al., 1989]: Let F be a continuous function on a bounded subset of D-dimensional space. Then there exists a two-layer network G with finite number of hidden units that approximates F arbitrarily well. For all \( x \) in the domain of F, \( |F(x) - G(x) | < \varepsilon \)

“a two-layer network can approximate any function”

Going from one to two layers dramatically improves the representation power of the network
How Deep Can They Be?

So many choices:
Architecture
# of hidden layers
# of units per hidden layer

Computational Issues:
Vanishing gradients
Gradients shrink as one moves away from the output layer
Convergence is slow

Opportunities:
Training deep networks is an active area of research
Layer-wise initialization (perhaps using unsupervised data)
Engineering: GPUs to train on massive labelled datasets
Some Results: Digit Classification

**Figure 11.10.** Architecture of the five networks used in the ZIP code example.

(similar to MNIST in A2, but not exactly the same)

**Figure 11.11.** Test performance curves, as a function of the number of training epochs, for the five networks of Table 11.1 applied to the ZIP code data.
Outline

Neural networks: non-linear classifiers

Learning weights: backpropagation of error

Autodifferentiation (in reverse mode)
Empirical Risk Minimization

Cross entropy loss
\[ \ell^{\text{xent}}(\overrightarrow{y^*}, y) = - \sum_k \overrightarrow{y^*}[k] \log p(y = k) \]

mean squared error/L2 loss
\[ \ell^{\text{L2}}(\overrightarrow{y^*}, y) = (\overrightarrow{y^*} - y)^2 \]

squared expectation loss
\[ \ell^{\text{sq-expt}}(\overrightarrow{y^*}, y) = |\overrightarrow{y^*} - p(y)|_2^2 \]

hinge loss
\[ \ell^{\text{hinge}}(\overrightarrow{y^*}, y) = \max \left\{ 0, 1 + \max_{j \neq y^*} (y[j] - \overrightarrow{y^*}[j]) \right\} \]
Gradient Descent: Backpropagate the Error

Set $t = 0$
Pick a starting value $\theta_t$
Until converged:

for example(s) $i$:

1. Compute loss $l$ on $x_i$
2. Get gradient $g_t = l'(x_i)$
3. Get scaling factor $\rho_t$
4. Set $\theta_{t+1} = \theta_t - \rho_t * g_t$
5. Set $t += 1$

**epoch**: a single run over all training data

**(mini-)batch**: a run over a subset of the data
Gradients for Feed Forward Neural Network

\[ y_k = \sigma \left( \beta_k^T \left( \sigma (w_j^T x + b_0) \right)_j \right) \]

\[ \mathcal{L} = -\sum_k \overrightarrow{y^*}[k] \log y_k \]

\[ \frac{\partial \mathcal{L}}{\partial \beta_{kj}} = -\frac{1}{y_{y^*}} \frac{\partial y_{y^*}}{\partial \beta_{kj}} \]

\[ \frac{\partial \mathcal{L}}{\partial w_{jl}} \]
Gradients for Feed Forward Neural Network

\[ y_k = \sigma \left( \beta_k^T \left( \sigma (w_j^T x + b_0) \right)_j \right) \]

\[ L = - \sum_k \overrightarrow{y}^*[k] \log y_k \]

\[ \frac{\partial L}{\partial \beta_{kj}} = -1 \frac{\partial y^*_y}{\partial \beta_{kj}} = \frac{-\sigma' (\beta_y^T h)}{\sigma (\beta_y^T h)} \frac{\partial \beta_k^T h}{\partial \beta_{kj}} \]

\[ \frac{\partial L}{\partial w_{jl}} \]
Gradients for Feed Forward Neural Network

\[ y_k = \sigma \left( \beta_k^T \left( \sigma (w_j^T x + b_0) \right)_j \right) \]

\[ \mathcal{L} = - \sum_k y^*_k \log y_k \]

\[ h: \text{a vector} \]

\[ \frac{\partial \mathcal{L}}{\partial \beta_{kj}} = -1 \frac{\partial y^*_k}{\partial \beta_{kj}} = \frac{-\sigma'(\beta_{y^*h})}{\sigma(\beta_{y^*h})} \frac{\partial \beta_k^T h}{\partial \beta_{kj}} = \frac{-\sigma'(\beta_{y^*h})}{\sigma(\beta_{y^*h})} \frac{\partial \sum_j \beta_{y^*j} h_j}{\partial \beta_{kj}} \]

\[ \frac{\partial \mathcal{L}}{\partial w_{jl}} \]
Gradients for Feed Forward Neural Network

\[ y_k = \sigma \left( \beta_k^T \left( \sigma \left( w_j^T x + b_0 \right) \right)_j \right) \]

\[ \mathcal{L} = - \sum_k \overrightarrow{y^*}[k] \log y_k \]

\[ h: \text{ a vector} \]

\[ \frac{\partial \mathcal{L}}{\partial \beta_{kj}} = \frac{-1}{y_j^*} \frac{\partial y_j^*}{\partial \beta_{kj}} = -\sigma'(\beta_y^T h) \frac{\partial \beta_k^T h}{\partial \beta_{kj}} = -\sigma'(\beta_y^T h) \frac{\partial \sum_j \beta_y^* h_j}{\partial \beta_{kj}} \]

\[ = \left( 1 - \sigma(\beta_y^T h) \right) h_j \]

\[ \frac{\partial \mathcal{L}}{\partial w_{jl}} = \left( 1 - \sigma(\beta_y^T h) \right) \left( \beta_y^* \sigma'(w_j^T x) x_l \right) \]
Gradients for Feed Forward Neural Network

\[ y_k = \sigma \left( \beta_k^T \left( \sigma \left( w_j^T x + b_0 \right) \right)_j \right) \]

\[ L = - \sum_k y^*_k \log y_k \]

\[ \frac{\partial L}{\partial \beta_{kj}} = \left( 1 - \sigma(\beta_{y^*}^T h) \right) h_j \]

\[ \frac{\partial L}{\partial w_{jl}} = \left( 1 - \sigma(\beta_{y^*}^T h) \right) (\beta_{y^*j} \sigma'(w_j^T x) x_l) \]
Outline

Neural networks: non-linear classifiers

Learning weights: backpropagation of error

Autodifferentiation (in reverse mode)
Finding Gradients

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

*what are the partial derivatives?*
Finding Gradients

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ \frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 + a(x_1 - x_2)^{a-1} - \frac{2x_1}{x_1^2 + x_2^2} \]
Finding Gradients

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ \frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 + a(x_1 - x_2)^{a-1} - \frac{2x_1}{x_1^2 + x_2^2} \]

\[ \frac{\partial f(x_1, x_2)}{\partial x_2} = -a(x_1 - x_2)^{a-1} - \frac{2x_2}{x_1^2 + x_2^2} \]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

“straight line” program
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

"straight line" program

computation graph
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[
\begin{align*}
z_1 &= x_1^2 \\
z_2 &= x_2^2 \\
z_3 &= (x_1 - x_2) \\
z_4 &= z_3^a \\
z_5 &= z_1 + z_2 \\
z_6 &= \log z_5 \\
z_7 &= z_1 + z_4 - z_6 \\
y &= z_7
\end{align*}
\]

“straight line” program
Autodifferentiation

\[
f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2)
\]

\[
\begin{align*}
z_1 &= x_1^2 \\
z_2 &= x_2^2 \\
z_3 &= (x_1 - x_2) \\
z_4 &= z_3^a \\
z_5 &= z_1 + z_2 \\
z_6 &= \log z_5 \\
z_7 &= z_1 + z_4 - z_6 \\
y &= z_7
\end{align*}
\]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

- \( z_1 = x_1^2 \)
- \( z_2 = x_2^2 \)
- \( z_3 = (x_1 - x_2) \)
- \( z_4 = z_3^a \)
- \( z_5 = z_1 + z_2 \)
- \( z_6 = \log(z_5) \)
- \( z_7 = z_1 + z_4 - z_6 \)
- \( y = z_7 \)

\[ \delta t = \frac{\partial y}{\partial t} \]

adjoint

goals:

\[ \frac{\partial y}{\partial x_1} \]

\[ \frac{\partial y}{\partial x_2} \]

\( \delta y = 1 \)
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[
\begin{align*}
    z_1 &= x_1^2 \\
    z_2 &= x_2^2 \\
    z_3 &= (x_1 - x_2) \\
    z_4 &= z_3^a \\
    z_5 &= z_1 + z_2 \\
    z_6 &= \log z_5 \\
    z_7 &= z_1 + z_4 - z_6 \\
    y &= z_7
\end{align*}
\]

adjoint

\[
\begin{align*}
    \frac{\partial y}{\partial t} &= \frac{\partial y}{\partial t} \\
    \frac{\partial y}{\partial x_1} &= \frac{\partial y}{\partial x_1} \\
    \frac{\partial y}{\partial x_2} &= \frac{\partial y}{\partial x_2} \\
    \frac{\partial y}{\partial z_7} &= \frac{\partial y}{\partial z_7}
\end{align*}
\]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[
\begin{align*}
z_1 &= x_1^2 \\
z_2 &= x_2^2 \\
z_3 &= (x_1 - x_2) \\
z_4 &= z_3^a \\
z_5 &= z_1 + z_2 \\
z_6 &= \log(z_5) \\
z_7 &= z_1 + z_4 - z_6 \\
y &= z_7
\end{align*}
\]

goals:
\[
\begin{align*}
\frac{\partial y}{\partial x_1} \\
\frac{\partial y}{\partial x_2}
\end{align*}
\]

adjoint
\[
\dot{t} = \frac{\partial y}{\partial t}
\]

\[
\begin{align*}
\dot{z}_7 &= \frac{\partial y}{\partial z_7} = 1 \\
\dot{z}_6 &= \frac{\partial y}{\partial z_6} = \frac{\partial y}{\partial z_7} \frac{\partial z_7}{\partial z_6} = \dot{z}_7 * -1
\end{align*}
\]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

goals:
\[ \frac{\partial y}{\partial x_1} \]
\[ \frac{\partial y}{\partial x_2} \]

\[ \delta t = \frac{\partial y}{\partial t} \]

adjoint

\[ \delta z_6 = \frac{\partial y}{\partial z_6} = \frac{\partial y}{\partial z_7} \frac{\partial z_7}{\partial z_6} = \delta z_7 \times -1 \]
\[ \delta z_4 = \frac{\partial y}{\partial z_4} = \frac{\partial y}{\partial z_7} \frac{\partial z_7}{\partial z_4} = \delta z_7 \times 1 \]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

goals:
\[ \frac{\partial y}{\partial x_1} \]
\[ \frac{\partial y}{\partial x_2} \]
\[ \frac{\partial z_5}{\partial z_5} = \frac{\partial y}{\partial z_7} \frac{\partial z_7}{\partial z_5} = \frac{\partial y}{\partial z_7} \frac{\partial z_7}{\partial z_6} \frac{\partial z_6}{\partial z_5} \]

adjoint

\[ \delta t = \frac{\partial y}{\partial t} \]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

adjoint goals:

\[ \frac{\partial y}{\partial x_1} \]
\[ \frac{\partial y}{\partial x_2} \]

\[ \frac{\partial}{\partial z_5} \]
\[ \frac{\partial}{\partial z_7} \]

\[ \frac{\partial}{\partial z_6} \]
\[ \frac{\partial}{\partial z_4} \]

\[ \frac{\partial}{\partial z_7} \]

\[ \frac{\partial}{\partial y} \]

\[ \frac{\partial}{\partial t} \]

\[ \frac{\partial y}{\partial t} = 1 \]
\[ \frac{\partial z_7}{\partial z_7} = 1 \]
\[ \frac{\partial z_6}{\partial z_7} = -1 \]
\[ \frac{\partial z_4}{\partial z_7} = 1 \]

\[ \frac{\partial z_5}{\partial z_5} = \frac{\partial y}{\partial z_5} = \frac{\partial y}{\partial z_7} \frac{\partial z_7}{\partial z_5} = \frac{\partial y}{\partial z_7} \frac{\partial z_7}{\partial z_6} \frac{\partial z_6}{\partial z_5} = \frac{\partial z_6}{z_5} * 1 \]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[
\begin{align*}
z_1 &= x_1^2 \\
z_2 &= x_2^2 \\
z_3 &= (x_1 - x_2) \\
z_4 &= z_3^a \\
z_5 &= z_1 + z_2 \\
z_6 &= \log z_5 \\
z_7 &= z_1 + z_4 - z_6 \\
y &= z_7
\end{align*}
\]

goals:

\[
\begin{align*}
\frac{\partial y}{\partial x_1} &=\frac{\partial y}{\partial z_1} = \frac{\partial y}{\partial z_7} \frac{\partial z_7}{\partial z_1} + \frac{\partial y}{\partial z_7} \frac{\partial z_7}{\partial z_6} \frac{\partial z_6}{\partial z_5} \frac{\partial z_5}{\partial z_1} \\
\frac{\partial y}{\partial x_2} &= \frac{\partial y}{\partial z_7} = 1 \\
\frac{\partial y}{\partial t} &= \frac{\partial y}{\partial t} = 1 \\
\frac{\delta z_7}{\delta z_7} &= 1 \\
\frac{\delta z_6}{\delta z_7} &= \frac{\delta z_7}{-1} \\
\frac{\delta z_4}{\delta z_7} &= \frac{\delta z_7}{1} \\
\frac{\delta z_5}{\delta z_6} &= \frac{\delta z_6}{\frac{1}{z_5}} 
\end{align*}
\]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

goals:
\[ \frac{\partial y}{\partial x_1} \]
\[ \frac{\partial y}{\partial x_2} \]

adjoint

\[ \delta t = \frac{\partial y}{\partial t} \]

\[ \delta z_1 = \frac{\partial y}{\partial z_1} = \delta z_7 \cdot 1 + \delta z_5 \cdot 1 \]

\[ \delta z_5 = \delta z_6 \cdot \frac{1}{z_5} \]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

goals:
\[ \frac{\partial y}{\partial x_1} \]
\[ \frac{\partial y}{\partial x_2} \]

\[ \delta t = \frac{\partial y}{\partial t} \]

adjoint

\[ \delta y = 1 \]
\[ \delta z_7 = \frac{\partial y}{\partial z_7} = 1 \]
\[ \delta z_6 = \delta z_7 \times -1 \]
\[ \delta z_4 = \delta z_7 \times 1 \]
\[ \delta z_5 = \delta z_6 \times \frac{1}{z_5} \]
\[ \delta z_1 + = \delta z_7 \times 1 \]
\[ \delta z_1 + = \delta z_5 \times 1 \]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

goals:
\[ \frac{\partial y}{\partial x_1} \]
\[ \frac{\partial y}{\partial x_2} \]

\[ \frac{\partial y}{\partial t} = \frac{\partial y}{\partial t} \]
adjoint

\[ \delta t = \frac{\partial y}{\partial t} \]

\[ \delta z_2 = \frac{\partial y}{\partial z_2} = \delta z_5 \times 1 \]

\[ \delta z_7 = \frac{\partial y}{\partial z_7} = 1 \]
\[ \delta z_6 = \delta z_7 \times -1 \]
\[ \delta z_4 = \delta z_7 \times 1 \]
\[ \delta z_5 = \delta z_6 \times \frac{1}{z_5} \]
\[ \delta z_1 += \delta z_7 \times 1 \]
\[ \delta z_1 += \delta z_5 \times 1 \]
Autodifferentiation

\[
f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2)
\]

\[
z_1 = x_1^2
\]

\[
z_2 = x_2^2
\]

\[
z_3 = (x_1 - x_2)
\]

\[
z_4 = z_3^a
\]

\[
z_5 = z_1 + z_2
\]

\[
z_6 = \log z_5
\]

\[
z_7 = z_1 + z_4 - z_6
\]

\[
y = z_7
\]

\[
\begin{align*}
\delta z_2 &= \delta z_5 \times 1 \\
\delta z_7 &= \frac{\partial y}{\partial z_7} = 1 \\
\delta z_6 &= \delta z_7 \times -1 \\
\delta z_4 &= \delta z_7 \times 1 \\
\delta z_5 &= \delta z_6 \times \frac{1}{z_5} \\
\delta z_1 &= \delta z_7 \times 1 \\
\delta z_1 &= \delta z_5 \times 1 \\
\end{align*}
\]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

goals:

\[ \frac{\partial y}{\partial x_1} \]
\[ \frac{\partial y}{\partial x_2} \]

\[ \frac{\partial y}{\partial t} = \frac{\partial y}{\partial t} \]

adjoint

\[ \delta t = \frac{\partial y}{\partial t} \]

\[ \delta z_7 = \frac{\partial y}{\partial z_7} = 1 \]
\[ \delta z_6 = \delta z_7 * -1 \]
\[ \delta z_4 = \delta z_7 * 1 \]
\[ \delta z_5 = \delta z_6 * \frac{1}{z_5} \]
\[ \delta z_1 = \delta z_7 * 1 \]
\[ \delta z_1 = \delta z_5 * 1 \]

\[ \delta x_1 += \delta z_1 * 2x_1 \]
\[ \delta x_1 += \delta z_3 * 1 \]
\[ \delta x_2 += \delta z_2 * 2x_2 \]
\[ \delta x_2 += \delta z_3 * -1 \]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[
\begin{align*}
    z_1 &= x_1^2 \\
    z_2 &= x_2^2 \\
    z_3 &= (x_1 - x_2) \\
    z_4 &= z_3^a \\
    z_5 &= z_1 + z_2 \\
    z_6 &= \log z_5 \\
    z_7 &= z_1 + z_4 - z_6 \\
    y &= z_7
\end{align*}
\]

goals:

\[
\begin{align*}
    \frac{\partial y}{\partial x_1} &= \frac{\partial x_1}{\partial z_1} + \frac{\partial x_2}{\partial z_1} + \frac{\partial x_1}{\partial z_3} + \frac{\partial x_2}{\partial z_3} \\
    \frac{\partial y}{\partial x_2} &= \frac{\partial x_1}{\partial z_7} + \frac{\partial x_2}{\partial z_7}
\end{align*}
\]

adjoint

\[
\frac{\partial t}{\partial y} = \frac{\partial y}{\partial t}
\]
Autodifferentiation

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

goals:
\[
\frac{\partial y}{\partial x_1} = \delta z_1 + \delta z_7 
\]
\[
\frac{\partial y}{\partial x_2} = \delta z_2 + \delta z_4 + \delta z_7 
\]

\[ \frac{\partial y}{\partial t} = \delta z_7 \]

autodifferentiation in reverse mode
Autodifferentiation in Reverse Mode

\[ f(x_1, x_2) = x_1^2 + (x_1 - x_2)^a - \log(x_1^2 + x_2^2) \]

\[ z_1 = x_1^2 \]
\[ z_2 = x_2^2 \]
\[ z_3 = (x_1 - x_2) \]
\[ z_4 = z_3^a \]
\[ z_5 = z_1 + z_2 \]
\[ z_6 = \log z_5 \]
\[ z_7 = z_1 + z_4 - z_6 \]
\[ y = z_7 \]

\[ \frac{\partial y}{\partial x_1} = \frac{\delta z_1 + \delta z_3 * 1}{x_1} \]
\[ \frac{\partial y}{\partial x_2} = \frac{\delta z_2 + \delta z_4 * 2x_2}{x_2} \]

goals:
\[ \frac{\partial y}{\partial x_1} \]
\[ \frac{\partial y}{\partial x_2} \]

\[ x_1 = 2 \quad x_2 = 1 \quad a = 1 \]
\[ f(x_1 = 2, x_2 = 1) \approx 3.390562 \]
\[ \nabla_x = (4.2, -1.4) \quad \text{by exact gradients} \]
\[ \nabla_x = (4.2, -1.4) \quad \text{by autodiff} \]
Code Proof of Autodiff

```python
def autodiff(x1, x2, a=1.0):
    z1 = x1**2
    z2 = x2**2
    z3 = (x1 - x2)
    z4 = z3**a
    z5 = z1 + z2
    z6 = numpy.log(z5)
    z7 = z1 + z4 - z6
    y = z7
    dy = 1
    dz7 = dy
    dz6 = dz7 * -1.0
    dz5 = dz6 * 1.0 / z5
    dz4 = dz7 * 1.0
    dz3 = dz4 * a * z3 ** (a - 1)
    dz2 = dz5 * 1.0
    dz1 = dz7 * 1.0 + dz5 * 1.0
    dx1 = dz1 * 2 * x1 + dz3 * 1.0
    dx2 = dz2 * 2 * x2 + dz3 * -1.0
    return dx1, dx2
```

```python
>> autodiff(2, 1)
(4.2, -1.4)
```
>> def f(x1, x2):
    return x1**2 + (x1-x2)**2 - numpy.log(x1**2+x2**2)

>> def autodiff(x1, x2, a=1.0):
    z1 = x1**2
    z2 = x2**2
    z3 = (x1-x2)
    z4 = z3**a
    z5 = z1 + z2
    z6 = numpy.log(z5)
    z7 = z1 + z4 - z6
    y = z7
    dy = 1
    dz7 = dy
    dz6 = dz7 * -1.0
    dz5 = dz6 * 1.0 / z5
    dz4 = dz7 * 1.0
    dz3 = dz4 * a * z3**a
    dz2 = dz5 * 1.0
    dz1 = dz7 * 1.0 + dz5 * 1.0
    dx1 = dz1 * 2 * x1 + dz3 * 1.0
    dx2 = dz2 * 2 * x2 + dz3 * -1.0
    return dx1, dx2

>> autodiff(2,1)
(4.2, -1.4)
Outline

Neural networks: non-linear classifiers

Learning weights: backpropagation of error

Autodifferentiation (in reverse mode)