Sensitivity of Gradient Accuracy for Deep Declarative Networks
Deep Declarative Networks?
Deep Declarative Networks (DDNs)

- New class of end-to-end learnable model
Deep Declarative Networks (DDNs)

- New class of end-to-end learnable model
- Nodes are defined as optimisation problem
Deep Declarative Networks (DDNs)

- New class of end-to-end learnable model
- Nodes are defined as optimisation problem
- Backpropagation by implicit differentiation
Differences

Standard Node

\[ y = \tilde{f}(x; \theta) \]

\[ \theta \quad D J(\theta) \]

\[ x \quad D J(x) \]

\[ y \quad D J(y) \]

Declarative Node

\[ y \in \arg\min_{u \in C} f(x, u; \theta) \]

\[ \theta \quad D J(\theta) \]

\[ x \quad D J(x) \]

\[ y \quad D J(y) \]

Diagrams from Deep Declarative Networks: A New Hope by Gould et al.
Differences

**Standard Node**

\[ y = \frac{1}{n} \sum_{i=1}^{n} \theta^T x_i \]

**Declarative Node**

\[ y \in \arg\min_{u \in \mathcal{C}} \left| u - \left( \frac{1}{n} \sum_{i=1}^{n} \theta^T x_i \right) \right| \]
Differences

Standard Node

Declarative Node

\[ y = \hat{f}(x; \theta) \]

\[ x \quad \text{D}J(x) \quad y \]

\[ \theta \quad \text{D}J(\theta) \]

\[ y \in \arg \min_{u \in C} f(x, u; \theta) \]

\[ x \quad \text{D}J(x) \quad y \]

\[ \theta \quad \text{D}J(\theta) \]

\[ y \in \arg \min_u \frac{1}{n} \left( \sum_{i=1}^{n} \sqrt{1 + (u - \theta^T x_i)^2} - 1 \right) \]
Problem

• Implicit function theorem (IFT) is used to compute gradient for backward pass
Problem

- Implicit function theorem (IFT) is used to compute gradient for backward pass

- IFT assumes that the optimisation problem is solved exactly
Implicit Function Theorem (IFT)

Unconstrained Case:

For $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, given

$$y(x) \in \arg \min_{u \in \mathbb{R}^m} f(x, u).$$

then as $\left[ \frac{\partial f}{\partial u}(x, y(x)) \right]_{1 \times m} = 0$, by the IFT

$$\left[ \frac{\partial y(x)}{\partial x} \right]_{m \times n} = - \left[ \frac{\partial^2 f(x, y(x))}{\partial u^2} \right]_{m \times m}^{-1} \left[ \frac{\partial^2 f(x, y(x))}{\partial x \partial u} \right]_{m \times n}$$

So we have a way to compute $\frac{\partial y(x)}{\partial x}$ but it requires computing the exact solution $y(x)$. 
Aim

How much does the gradient signal degrade when the optimisation problem is not solved exactly
Why

Good algorithm can trade-off optimality for speed
Why

Provide guarantees for gradients if global optimum can’t be found
Design

• Robust Pooling

• $L_p$ Projection
Robust Pooling

Data

Robust Pooling Node
\[ y \in \text{argmin}_u f(x, u) \]

Mean Squared Objective
\[ J = \frac{1}{2} \| y \|_2^2 \]

Iterative Solver
\[ \text{argmin}_u f(x, u) \]

Outer Optimisation

Inner Optimisation

\[ \frac{Dy}{Dz} \]

\[ \frac{DJ}{Dy} \]
Robust Pooling
Robust Pooling animations for optimal pooling

Convex

Non-Convex
Robust Pooling animations for suboptimal pooling

Convex

Non-Convex
Results

Convex functions (pseudo-huber & huber) are robust

Non-convex functions (welsch & trunc-quad) are sensitive
\( L_p \) Projection

\[
y_p \in \arg \min_{u \in \mathbb{R}^n} \frac{1}{2} \| u - x \|_2^2
\]
subject to \( \| u \|_p = 1 \)

Diagrams from Deep Declarative Networks: A New Hope by Gould et al.
Results - $L_2$ Projection

Frobenius Norm

Cosine Similarity
Results - $L_1$ Projection
Results - $L_\infty$ Projection
Future Work

• Use simpler gradient descent
Future Work

- Use simpler gradient descent

- How well these results apply to
  - Other constrained and unconstrained problems
  - actual DDNs
Any Questions?