Modular Learning

John Langford

For COMS-4771
Real Learning Systems are complicated

How do we learn the parameters of the grass predictor?
Outline

1. Subproblem Learning

2. End-to-End Learning

3. Extensions of Gradient Descent
Subproblem Learning is Powerful

Theorem: Assuming AES encryption is unbreakable, there exists learning problems $D$ for which direct learning of subproblems $D_1, ..., D_m$ is tractable, yet learning without subproblems is computationally intractable.

In other words: learning the full problem can be hard, but if you know the right subproblems to solve, it can become easy.
Proof: Let $D$ be a distribution on $x = \text{AES encrypted IMs}$ and $y = \text{plain text IMs}$.

1. $y$ is essentially unpredictable given $x$.

2. But AES can be written as a circuit of and/or/not gates.

A circuit of simple gates

and “and”, “or”, and “not” are all learnable.
A problem with Subproblem Learning

Theorem: (Independent Learning Weakness) For any $m$, there exists a learning problem $D$ with subproblems $D_1, ..., D_m$ such that:

$$e(C_f, D) = \sum_i e(f_i, D_i)$$

where $e(C_f, D)$ is the error rate of the circuit $C$ composed of the learned $f_1, ..., f_m$.

Proof: Create a circuit where any error at any gate implies an overall error.

Implication: erring on any subproblem can cause an overall error.
Outline

1. Subproblem Learning

2. End-to-End Learning

3. Extensions of Gradient Descent
End-to-End learning

Essential idea: do a joint optimization of all subproblems to improve performance.

Primary method: gradient descent
A Simplification of the Problem

Function to learn $= g(w_g, f(w_f, x))$

Suppose we care about squared loss: $E_{x,y \sim D} (g(w_g, f(w_f, x)) - y)^2$

How should we tune $w_g$ and $w_f$?
Gradient Descent

\[-\frac{\partial}{\partial w_g} (g(w_g, f(w,f,x)) - y)^2\]

\[= 2(g(w_g, f(w,f,x)) - y) \frac{\partial g(w_g,f(w,f,x))}{\partial w_g}\]

and

\[-\frac{\partial}{\partial w_f} (g(w_g, f(w,f,x)) - y)^2\]

\[= 2(g(w_g, f(w,f,x)) - y) \frac{\partial g(w_g,f(w,f,x))}{\partial w_f}\]

\[= 2(g(w_g, f(w,f,x)) - y) \frac{\partial g(w_g,f(w,f,x))}{\partial f(w,f,x)} \frac{\partial f(w,f,x)}{\partial w_f}\]

“Chain rule of differentiation”
Gradient chain rule goes in the opposite direction to evaluation.
Some notes about chain rule learning

- Needs continuous functions.

- In general, local minima bite, unless the function is convex.

- Sigmoid $h(x) = \frac{1}{1 + e^{-x}}$ is convenient: $\frac{\partial h(x)}{\partial x} = h(x)(1 - h(x))$

- Since derivatives are linear, if $g$ uses $f$ twice (happens all the time in a big circuit), the updates to $w_f$ sum.

- The update can be online.
• The derivative on $w_f$ can collapse to zero very quickly with depth of a circuit. Most people use shallow structures (See Yann LeCun’s Convolutional Neural Networks for a nonshallow network).
Outline

1. Subproblem Learning

2. End-to-End Learning

3. Extensions of Gradient Descent
Problem: Your derivative is discontinuous

Solution: Ignore the problem—you never land on the discontinuity in practice.

(See the study of “subgradients”.)
Problem: a set of weights must sum to 1

Solution:

1. compute a derivative

2. gradient descent step

3. project back into the allowed set

(See “extragradient” for more details.)
Problem: function is not differentiable at all

Solution:

1. Try computing a discrete gradient: test how small changes in input alter output. Treat the discrete gradient as a gradient.

2. Find some approximation which is differentiable.
General Strategy for coping with Modular learning problems

1. Take advantage of all subproblem knowledge you have first.

2. Apply (extra|sub|discrete)gradient for final tuning.