Game theory for Neural Networks

C³ Research Camp

David Balduzzi
VUW
Goals

“a justification is objective if, in principle, it can be tested and understood by anybody”
— Karl Popper
Goals

Objective knowledge

• How is knowledge created?

• How is knowledge shared?
Background

• Deductive reasoning (*a priori* knowledge)
  • foundation of mathematics
  • formal: mathematical logic
  • operational: Turing machines, lambda calculus
Tiny Turing Machine

Alvy Ray Smith, based on [Yurii Rogozhin 1996]
Tiny Turing Machine

Alvy Ray Smith, based on [Yurii Rogozhin 1996]
· **Deductive reasoning** (*a priori* knowledge)
  - foundation of mathematics
  - **formal**: mathematical logic
  - **operational**: Turing machines, lambda calculus

· **Inductive reasoning** (*empirical* knowledge)
  - foundation of science
  - **formal**: learning theory
  - **operational**: SVMs, neural networks, etc.
• What’s missing?
  • To be objective, knowledge must be shared
  • How can agents *share* knowledge?

• A theory of *distributed reasoning*
  • induction builds on deduction
  • distribution should build on both
Neural Networks

- Superhuman performance on object recognition (ImageNet)
- Outperformed humans at recognising street-signs (Google streetview).
- Superhuman performance on Atari games (Google).
- Real-time translation: English voice to Chinese voice (Microsoft).
30 years of trial-and-error

- Architecture
  - Weight-tying (convolutions)
  - Max-pooling
- **Nonlinearity**
  - Rectilinear units (Jarrett 2009)
- Credit assignment
  - Backpropagation
- Optimization
  - Stochastic Gradient Descent
  - Nesterov momentum, BFGS, AdaGrad, etc.
  - RMSProp (Riedmiller 1993, Hinton & Tieleman 200x)
- Regularization
  - Dropout (Srivastava 2014) or
  - Batch-Normalization (Szegedy & Ioffe 2015)
This tutorial

- **Concrete problem:**
  - Why does gradient descent work on neural nets?
  - Not just gradient descent; many methods designed for convex problems work surprisingly well on neural nets.

- **Big picture:**
  - Neural networks are populations of learners
  - that distribute knowledge extremely effectively.

We should study them as such.
Outline

• Deduction
• Induction
• Gradients
• Games
• Neural networks
Deductive reasoning

“we are regarding the function of the mathematician as simply to determine the truth or falsity of propositions”
Deductive reasoning

All men are mortal
Socrates is a man

Socrates is mortal
Deductive reasoning

All men are mortal
Socrates is a man

Socrates is mortal

Idea: Intelligence is deductive reasoning!
A Proposal for the

DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE

June 17 - Aug. 16

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.
50 years later

Trenchard More, John McCarthy, Marvin Minsky, Oliver Selfridge, Ray Solomonoff
A bump in the road

“To understand the real world, we must have a different set of primitives from the relatively simple line trackers suitable and sufficient for the blocks world”

— Patrick Winston (1975)
Director of MIT’s AI lab from 1972-1997
The AI winter

http://en.wikipedia.org/wiki/AI_winter

- 1966: the failure of machine translation,
- 1970: the abandonment of connectionism,
- 1971–75: DARPA's frustration with the Speech Understanding Research program at Carnegie Mellon University,
- 1973: the large decrease in AI research in the United Kingdom in response to the Lighthill report,
- 1973–74: DARPA's cutbacks to academic AI research in general,
- 1987: the collapse of the Lisp machine market,
- 1988: the cancellation of new spending on AI by the Strategic Computing Initiative,
- 1993: expert systems slowly reaching the bottom,
- 1990s: the quiet disappearance of the fifth-generation computer project's original goals,
“Intelligence is 10 million rules”
— Doug Lenat

Cyc is an artificial intelligence project that attempts to assemble a comprehensive ontology and knowledge base of everyday common sense knowledge, with the goal of enabling AI applications to perform human-like reasoning.

Reductio ad absurdum
Deductive reasoning

• Structure:
  • axioms;
  • logical laws;
  • production system

• Derive complicated (empirical) truths from simple (self-evident) truths
• Deductive reasoning provides no guidance about which axiom(s) to eliminate when “derived truths” contradict experience
• In general, deductive reasoning has nothing to say about mistakes except “avoid them”
• In retrospect, it’s hard to understand why anyone thought deduction would yield artificial intelligence.
Inductive reasoning

“this book is composed [...] upon one very simple theme [...] that we can learn from our mistakes”
Inductive reasoning

Socrates is mortal
Plato is mortal
[ ... more examples ... ]

All men are mortal
Inductive reasoning

All observed As are Bs
a is an A

a is a B
The problem of induction

All observed As are Bs
a is an A

a is a B

This is clearly false. The conclusion does not follow from the premises.
“If a machine is [...] infallible, it cannot also be intelligent. There are [...] theorems which say almost exactly that. But these theorems say nothing about how much intelligence may be displayed if a machine makes no pretence at infallibility.”

[ it’s ok to make mistakes ]
Let’s look at some examples
Sequential prediction

**Scenario:** At time $t$, Forecaster predicts 0 or 1. Nature then reveals the truth.

Forecaster has access to $N$ experts. One of them is always correct.

**Goal:** Predict as accurately as possible.
Halving Algorithm

Set $t = 1$.

While $t > 0$:

Step 1. Predict by majority vote.

Step 2. Remove experts that are wrong.

Step 3. $t \leftarrow t + 1$
While $t > 0$:

Step 1. Predict by majority vote.

Step 2. Remove experts that are wrong.

Step 3. $t \leftarrow t+1$

Question: How long to find correct expert?
Halving Algorithm

Set $t = 1$.

While $t > 0$:

Step 1. Predict by majority vote.

Step 2. Remove experts that are wrong.

Step 3. $t \leftarrow t + 1$

BAD!!! How long to find correct expert?
Halving Algorithm

Set $t = 1$.

While $t > 0$:

Step 1. Predict by majority vote.

Step 2. Remove experts that are wrong.

Step 3. $t \leftarrow t + 1$

Question: How many errors?
Halving Algorithm

**Step 1.** Predict by majority vote.

**Step 2.** Remove experts that are wrong.

How many errors?
Halving Algorithm

When algorithm makes a mistake,

Step 1. Predict by majority vote.

Step 2. Remove experts that are wrong.

it removes \( \geq \frac{1}{2} \) half of experts

How many errors?
Halving Algorithm

When algorithm makes a mistake,

Step 1. Predict by majority vote.
Step 2. Remove experts that are wrong.

It removes ≥ half of experts

How many errors? \( \leq \log N \)
What's going on?

Didn't we just use deductive reasoning!?!
What's going on?

Didn't we just use deductive reasoning!?!?

Yes... but No!
What's going on?

**Algorithm:** makes educated guesses about Nature
(inductive)

**Analysis:** proves theorem about number of errors
(deductive)
What’s going on?

Algorithm: makes educated guesses about Nature (inductive)

Analysis: proves theorem about number of errors (deductive)

The algorithm learns — but it does not deduce!
Adversarial prediction

Scenario: At time $t$, Forecaster predicts 0 or 1. Nature then reveals the truth.

Forecaster has access to $N$ experts. One of them is always correct. Nature is adversarial.

Goal: Predict as accurately as possible.
At time $t$, Forecaster predicts 0 or 1. Nature then reveals the truth. Forecaster has access to $N$ experts. One of them is always correct. Nature is adversarial.

**Goal:** Predict as accurately as possible. Seriously?!?!
Let $m^*$ be the best expert in hindsight.

\[ \text{regret} := \# \text{ mistakes(Forecaster)} - \# \text{ mistakes}(m^*) \]

Goal: Predict as accurately as possible. Minimize regret.
Halving Algorithm

Set \( t = 1 \).

While \( t > 0 \):

Step 1. Predict by majority vote.

Step 2. Remove experts that are wrong.

Step 3. \( t \leftarrow t+1 \)

Question: What is the regret?
Halving Algorithm

Set \( t = 1 \).

While \( t > 0 \):

Step 1. Predict by majority vote.

Step 2. Remove experts that are wrong.

Step 3. \( t \leftarrow t + 1 \).

ADVERSARIAL SETTING

Question: What is the regret?
Weighted Majority Alg

Set $t = 1$. Pick $\beta$ in $(0,1)$. Assign 1 to experts.

While $t \leq T$: Predict by weighted majority vote.

Step 1. Multiply incorrect experts by $\beta$.

Step 2. $t \leftarrow t+1$

Step 3.

Question: What is the regret?
Weighted Majority Alg

Set $t = 1$.

While $t \leq T$:

Step 1. Predict by weighted majority vote.

Step 2. Multiply incorrect experts by $\beta$.

Step 3. $t \leftarrow t+1$

What is the regret? $\leq \sqrt{\frac{T \cdot \log N}{2}}$ [choose $\beta$ carefully]

Pick $\beta$ in $(0,1)$. Assign 1 to experts.
Loss functions

• 0/1 loss

\[ \ell(y, y') = \begin{cases} 
0 & \text{if } y = y' \\
1 & \text{else}
\end{cases} \]

• Mean square error

\[ \ell(y, y') = \frac{1}{2} (y - y')^2 \]

• Logistic loss

\[ \ell(p, i) = \begin{cases} 
- \log(1 - p) & \text{if } i = 0 \\
- \log(p) & \text{else}
\end{cases} \]
Loss functions

- 0/1 loss
  \[
  \ell(y, y') = \begin{cases} 
  0 & \text{if } y = y' \\
  1 & \text{else}
  \end{cases}
  \]

- Mean square error
  \[
  \ell(y, y') = \frac{1}{2} (y - y')^2
  \]

- Logistic loss
  \[
  \ell(p, i) = \begin{cases} 
  -\log(1 - p) & \text{if } i = 0 \\
  -\log(p) & \text{else}
  \end{cases}
  \]

not convex

convex
Exp. Weights Alg.

Set $t = 1$. Pick $\beta > 0$. Assign 1 to experts.

While $t \leq T$:

Step 1. Predict the weighted sum.

Step 2. Multiply expert $i$ by $e^{-\beta \cdot \ell(f_i^t, y^t)}$

Step 3. $t \leftarrow t+1$

What is the regret? $\leq \log N + \sqrt{2T \log N}$

[ choose $\beta$ carefully ]
Inductive reasoning

**Structure:**
- game played over a series of rounds
- on each round, **experts** make predictions
- Forecaster takes **weighted average** of predictions
- Nature reveals the truth and a **loss** is incurred
- Forecaster adjusts weights
  - correct experts $\rightarrow$ increase weights;
  - incorrect $\rightarrow$ decrease

**Goal:**
- average loss per round converges to best-in-hindsight
Inductive reasoning

• How it works
  • Forecaster adjusts strategy according to prior experience
  • Current strategy encodes previous mistakes
  • No guarantee on any particular example
  • Performance is optimal on average

• Still to come
  • How can Forecaster communicate its knowledge with others, in a way that is useful to them?
Gradient descent

“the great watershed in optimisation isn’t between linearity and nonlinearity, but convexity and nonconvexity” — Tyrrell Rockafellar
Gradient descent

• Everyone knows (?) gradient descent

• Take a fresh look:
  • gradient descent is a no-regret algorithm
  • → game theory
  • → neural networks
Online Convex Opt.

Scenario:
Convex set $\mathcal{K}$; differentiable loss $L(a,b)$ that is convex function of $a$

At time $t$,
Forecaster picks $a_t$ in $\mathcal{K}$
Nature responds with $b_t$ in $\mathcal{K}$

[ Nature is adversarial ]

Forecaster’s loss is $L(a,b)$

Goal:
Minimize regret.
Follow the Leader

Idea: Predict the $a_t$ that would have worked best on $\{ b_1, \ldots, b_{t-1} \}$
Follow the Leader

Set $t = 1$.

While $t \leq T$:

Step 1. \hspace{1cm} $t \leftarrow t+1$

Step 2. \hspace{1cm} $a_t := \arg\min_{a \in \mathcal{K}} \left[ \sum_{i=1}^{t-1} \mathcal{L}(a, b_i) \right]$

Idea: Predict the $a_t$ that would have worked best on $\{b_1, \ldots, b_{t-1}\}$
Follow the Leader BAD!

Problem: Nature pulls Forecaster back-and-forth
No memory!

Step 1. Set $t = 1$.

While $1 < t$:

Step 2. Pick $a_1$ at random.

$t \leftarrow t + 1$

$a_t := \arg\min_{a \in \mathcal{K}} \left[ \sum_{i=1}^{t-1} \mathcal{L}(a, b_i) \right]$
Set $t = 1$.

While $t \leq T$:

Step 1.

Step 2.

Pick $a_1$ at random.

$t \leftarrow t + 1$

$$a_t := \text{argmin}_{a \in \mathcal{A}} \left[ \sum_{i=1}^{t-1} \mathcal{L}(a, b_i) + \frac{\beta}{2} \cdot \|a\|_2^2 \right]$$

regularize
Set $t = 1$.

While $t \leq T$:

Step 1.

Step 2.

FTRL

Pick $a_1$ at random.

$t \leftarrow t + 1$

$a_t \leftarrow a_{t-1} - \beta \cdot \frac{\partial}{\partial a} \mathcal{L}(a_{t-1}, b_{t-1})$

gradient descent
Set $t = 1$.

While $t \leq T$: 

Step 1.

Step 2.

Pick $a_1$ at random.

$t \leftarrow t + 1$

$a_t \leftarrow a_{t-1} - \beta \cdot \frac{\partial}{\partial a} \mathcal{L}(a_{t-1}, b_{t-1})$

Intuition: $\beta$ controls memory
Set $t = 1$. 
While $t \leq T$: 
  
  **Step 1.** 
  Pick $a_1$ at random.
  
  **Step 2.** 
  $t \leftarrow t + 1$
  
  $a_t \leftarrow a_{t-1} - \beta \cdot \frac{\partial}{\partial a} \mathcal{L}(a_{t-1}, b_{t-1})$

What is the regret? 
$\leq \text{diam}(\mathcal{K}) \cdot \text{Lipschitz}(\mathcal{L}) \cdot \sqrt{T}$
[ choose $\beta$ carefully ]
Logarithmic regret

Convergence at rate $\sqrt{\frac{T}{T}}$ is actually quite slow

- For many common loss functions
  - (e.g. mean-square error, logistic loss)
- it is possible to do much better

\[
\text{regret} \leq \frac{\log T}{T}
\]

[ choose $\beta$ carefully ]
Information theory

Two connections

• Entropic regularization $\leftrightarrow$
  • Exponential Weights Algorithm

• Analogy between gradients and Fano information
  • “the information represented by $y$ about $x$”

$$\log \frac{P(x|y)}{P(x)}$$
Gradient descent

- Gradient descent is fast, cheap, and guaranteed to converge — even against adversaries
  - “a hammer that smashes convex nails”
- GD is well-understood in convex settings
- Convergence rate depends on the structure of the loss, size of search space, etc.
Convex optimization

Convex methods

• Stochastic gradient descent
• Nesterov momentum
• BFGS (Broyden-Fletcher-Goldfarb-Shanno)
• Conjugate gradient methods
• AdaGrad
Online Convex Opt.
(deep learning)

Apply **Gradient Descent** to **nonconvex** optimization (neural networks).

- Theorems don’t work (not convex)
- tons of engineering (decades of tweaking)

Amazing performance.
Deep Learning

- Superhuman performance on object recognition (ImageNet)
- Outperformed humans at recognising street-signs (Google streetview).
- Superhuman performance on Atari games (Google).
- Real-time translation: English voice to Chinese voice (Microsoft).
“An equilibrium is not always an optimum; it might not even be good. This may be the most important discovery of game theory.”

— Ivar Ekeland
Game theory

• So far
  • Seen how Forecaster can learn to use “expert” advice.
  • Forecaster modifies the weights of experts such that it converges to optimal-in-hindsight predictions.

• Who are these experts?
  • We will model them as Forecasters in their own right
  • First, let’s develop tools for modeling populations.
Setup

- Players A, B with actions \{a_1, a_2, \ldots a_m\} and \{b_1, b_2, \ldots b_n\}

- Two payoff matrices, describing the losses/gains of the players for every combination of moves

- Goal of players is to maximise-gain / minimise-loss

- Neither player knows what the other will do
Setup

• Players A, B with actions \{a_1, a_2, \ldots, a_m\} and \{b_1, b_2, \ldots, b_n\}

• Two payoff matrices, describing the losses/gains of the players for every combination of moves

**Rock-paper -scissors (zero-sum game)**

<table>
<thead>
<tr>
<th>A</th>
<th>R</th>
<th>P</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>P</td>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>R</th>
<th>P</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>P</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>S</td>
<td>-1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Minimax theorem

\[ \inf_{a \in \mathcal{K}} \sup_{b \in \mathcal{K}} \mathcal{L}(a, b) = \sup_{b \in \mathcal{K}} \inf_{a \in \mathcal{K}} \mathcal{L}(a, b) \]
Minimax theorem

\[ \inf_{a \in \mathcal{K}} \sup_{b \in \mathcal{K}} \mathcal{L}(a, b) = \sup_{b \in \mathcal{K}} \inf_{a \in \mathcal{K}} \mathcal{L}(a, b) \]

Forecaster picks \( a \),
Nature responds \( b \)
Minimax theorem

$$\inf_{a \in \mathcal{K}} \sup_{b \in \mathcal{K}} \mathcal{L}(a, b) = \sup_{b \in \mathcal{K}} \inf_{a \in \mathcal{K}} \mathcal{L}(a, b)$$

Forecaster picks $a$, Nature responds $b$

Nature picks $b$, Forecaster responds $a$
**Minimax theorem**

\[
\inf_{a \in \mathcal{K}} \sup_{b \in \mathcal{K}} \mathcal{L}(a, b) = \sup_{b \in \mathcal{K}} \inf_{a \in \mathcal{K}} \mathcal{L}(a, b)
\]

Forecaster picks \(a\), Nature responds \(b\)

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>P</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>P</td>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>

Nature picks \(b\), Forecaster responds \(a\)

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>P</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>P</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>S</td>
<td>-1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Minimax theorem

\[
\inf_{a \in \mathcal{A}} \sup_{b \in \mathcal{B}} \mathcal{L}(a, b) = \sup_{b \in \mathcal{B}} \inf_{a \in \mathcal{A}} \mathcal{L}(a, b)
\]

Forecaster picks \(a\),
Nature responds \(b\)

Nature picks \(b\),
Forecaster responds \(a\)

going first hurts Forecaster, so

\[
\inf_{a \in \mathcal{A}} \sup_{b \in \mathcal{B}} \mathcal{L}(a, b) \geq \sup_{b \in \mathcal{B}} \inf_{a \in \mathcal{A}} \mathcal{L}(a, b)
\]
Minimax theorem

Let $m^*$ be the best move in hindsight.

\[ \text{regret} := \text{loss(Forecaster)} - \text{loss}(m^*) \]

Proof idea: \[
\inf_{a \in \mathcal{A}} \sup_{b \in \mathcal{A}} \mathcal{L}(a, b) \leq \sup_{b \in \mathcal{A}} \inf_{a \in \mathcal{A}} \mathcal{L}(a, b) \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]
Minimax theorem

Let $m^*$ be the best move in hindsight.

**regret** := loss(Forecaster) - loss($m^*$)

**Proof idea:**

$$\inf_{a \in \mathcal{K}} \sup_{b \in \mathcal{K}} L(a, b) \leq \sup_{b \in \mathcal{K}} \inf_{a \in \mathcal{K}} L(a, b)$$

No-regret algorithm $\rightarrow$ Forecaster can asymptotically match hindsight
Minimax theorem

Let $m^*$ be the best move in hindsight.

$$\text{regret} := \text{loss(Forecaster)} - \text{loss}(m^*)$$

Proof idea:

$$\inf_{a \in \mathcal{K}} \sup_{b \in \mathcal{K}} L(a, b) \leq \sup_{b \in \mathcal{K}} \inf_{a \in \mathcal{K}} L(a, b)$$

No-regret algorithm $\rightarrow$ Forecaster can asymptotically match hindsight

$\rightarrow$ Order of players doesn’t matter asymptotically
Minimax theorem

Let $m^*$ be the best move in hindsight.

**regret** := loss(Forecaster) - loss($m^*$)

**Proof idea:**

No-regret algorithm $\rightarrow$ Forecaster can asymptotically match hindsight

$\rightarrow$ Order of players doesn’t matter asymptotically

$\bar{a} = \frac{1}{T} \sum_{t=1}^{T} a_t$

$\rightarrow$ Convert series of moves into average via online-to-batch.
Nash equilibrium

Nash equilibrium $\iff$ no player benefits by deviating
Nash equilibrium

Prisoner’s dilemma:

<table>
<thead>
<tr>
<th></th>
<th>Deny</th>
<th>Confess</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deny</td>
<td>-2/-2</td>
<td>-6/0</td>
</tr>
<tr>
<td>Confess</td>
<td>0/-6</td>
<td>-6/-6</td>
</tr>
</tbody>
</table>

Nash equilibrium ←→ no player benefits by deviating
Nash equilibrium

Problems:

• Computationally hard (PPAD-complete, Daskalakis 2008)

• Some outcomes are very bad
# Nash equilibrium

**Blind-intersection dilemma:**

<table>
<thead>
<tr>
<th></th>
<th>Stop</th>
<th>Go</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>0/0</td>
<td>10/0</td>
</tr>
<tr>
<td>Go</td>
<td>0/10</td>
<td>-∞/-∞</td>
</tr>
</tbody>
</table>
**Nash equilibrium**

**Blind-intersection dilemma:**

<table>
<thead>
<tr>
<th></th>
<th>Stop</th>
<th>Go</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>0/0</td>
<td>10/0</td>
</tr>
<tr>
<td>Go</td>
<td>0/10</td>
<td>-∞/-∞</td>
</tr>
</tbody>
</table>

**Equilibrium:** both drivers STOP; no-one uses intersection
Correlated equilibrium

Blind-intersection dilemma:

<table>
<thead>
<tr>
<th></th>
<th>Stop</th>
<th>Go</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>0/0</td>
<td>10/0</td>
</tr>
<tr>
<td>Go</td>
<td>0/10</td>
<td>-∞/-∞</td>
</tr>
</tbody>
</table>

Traffic light:

\[
P(x, y) = \begin{cases} 
\frac{1}{2} & \text{if } x = S \text{ and } y = G \\
\frac{1}{2} & \text{if } x = G \text{ and } y = S \\
0 & \text{else}
\end{cases}
\]
Correlated equilibrium

Blind-intersection dilemma:

- Better outcomes for players:
  Correlated equilibria *can* be much better than Nash equilibria (depending on the signal)
Convex games

- Players \( \{P_i\}_{i=1}^{N} \) with \textbf{convex} action sets \( \{K_i\}_{i=1}^{N} \)

- Loss functions \( \ell_i(w_1, \ldots, w_N) \) where \( w_i \in K_i \) and each loss is \textbf{separately convex} in each argument

- Players aim to minimize their losses

- Player do not know what other players will do
No-regret $\rightarrow$ Corr. eqm

Sketch of proof:

Since player $j$ applies a no-regret algorithm, it follows that after $T$ rounds, its sequence of moves \{\(w^t_j\)\} has a loss within $\varepsilon$ of the optimal, $w^*_j$.

$$\frac{1}{T} \sum \ell_i(w^t_j, w^{t-j}) - \ell_i(w^*_j, w^{t-j}) \leq \varepsilon \quad \forall w^*_j$$

Interpreting the historical sequence of plays as a signal obtains

$$\mathbb{E} \left[ \ell(w) \right] \leq \mathbb{E} \left[ \ell(w^*_j, w^{-j}) \right] + \varepsilon$$

The result follows by letting $\varepsilon \rightarrow 0$. 
From Nash equilibrium to correlated equilibrium:

- **Computationally tractable:**
  No-regret algorithms converge on coarse correlated equilibria

- **Better outcomes:**
  Signal is the past actions of players, chosen according to an optimal strategy
Game theory

- Every game has a Nash equilibrium
  - Economists typically assume that economy is in Nash equilibrium
  - But Nash equilibria are **hard** to find and are often **unpleasant**

- Correlated equilibria are easier to compute and **can** be better for the players

- **A signal is required to coordinate players:**
  - set signally externally
    - e.g. traffic light
  - in **repeated** games, **construct** signal as the average of historical behavior
    - e.g. no-regret learning
Game theory

• **Correlated equilibrium**
  • Players converge on mutually agreeable outcomes by adapting to each other’s behavior over repeated rounds
  • The signal provides a “strong hint” about where to find equilibria.

• **Nash equilibrium**
  • It’s hard to guess what others will do when playing *de novo* (intractability); and
  • You haven’t established a track record to build on (undesirability)
Gradients and Games

Gradient descent is a **no-regret algorithm**

- Insert players using gradient descent into any convex game, and they will converge on a correlated equilibrium

- Global convergence to correlated equilibrium is controlled by convergence of players to their best-in-hindsight local optimum

Next:
- Gradient descent + chain rule = backprop
Neural networks

“Anything a human can do in a fraction of a second, a deep network can do too” — Ilya Sutskever
History

• Origins
  • Backpropagation (Werbos 1974, Rumelhart 1986)
  • Convolutional nets (Fukushima 1980, LeCun 1990)

• Mid-1990s — Mid-2000s
  • back to convex methods; neural networks ignored

• Unsupervised pre-training
  • restricted Boltzmann machines (Hinton 2006)
  • autoencoders (Bengio 2007)

• Convolutional nets
  • sharp improvement on ImageNET challenge (Krizhevsky 2012)
  • back to fully-supervised training
• Convolutional nets
  • sharp improvement on ImageNET challenge (Krizhevsky 2012)
  • back to fully-supervised training
30 years of trial-and-error

- Architecture
  - Weight-tying (convolutions)
  - Max-pooling
- **Nonlinearity**
  - Rectilinear units (Jarrett 2009)
- Credit assignment
  - Backpropagation
- Optimization
  - Stochastic Gradient Descent
  - Nesterov momentum, BFGS, AdaGrad, etc.
  - RMSProp (Riedmiller 1993, Hinton & Tieleman 200x)
- Regularization
  - Dropout (Srivastava et al 2014) or
  - Batch-Normalization (Szegedy & Ioffe 2015)
Why (these) NNs?

• Why neural networks?
  • GPUs
  • Good local minima (no-one knows why)
  • “Mirrors the world’s compositional structure”

• Why ReLUs, Why Max-Pooling, Why convex methods?
  • “locally” convex optimization (this talk)
Neural networks

Input

$W_1$ Matrix mult

Layer$_1$

$W_2$ Matrix mult

Layer$_2$

$W_3$ Matrix mult

Output

$f_W(x)$
Neural networks

Input

Layer 1

Matrix mult

$W_1$

Nonlinearity

Layer 2

Matrix mult

$W_2$

Nonlinearity

Layer 3

Matrix mult

$W_3$

Output

$\ell(f_W(x), y)$

loss

label
Theorem (Leshno 1993): A two-layer neural network with infinite width and nonpolynomial nonlinearity is dense in the space of continuous functions.

\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

\[ \rho(x) = \max(0, x) \]

\[ \tau(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \]
The chain rule

\[ f \circ g \circ h(x) = f(g(h(x))) \]

\[ \frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{dx} \]
Linear networks

\[ W_1 \]

\[ W_2 \]

\[ W_3 \]
Feedforward sweep

\[ x_{\text{out}} = \left( \prod_{i=1}^{\#L} W_i \right) \cdot x_{\text{in}} \]
Weight updates

Gradient descent

\[ W_{ij}^{t+1} \leftarrow W_{ij}^t - \eta \cdot \frac{\partial \ell}{\partial W_{ij}} \]
Weight updates

Gradient descent

\[ W_{ij}^{t+1} \leftarrow W_{ij}^t - \eta \cdot \frac{\partial \ell}{\partial W_{ij}} \]

by the chain rule

\[ \frac{\partial \ell}{\partial W_{ij}} = \frac{\partial \ell}{\partial x_j} \cdot \frac{\partial x_j}{\partial W_{ij}} \]
Weight updates

Gradient descent

\[ W_{ij}^{t+1} \leftarrow W_{ij}^t - \eta \cdot \frac{\partial \ell}{\partial W_{ij}} \]

by the chain rule

\[ \frac{\partial \ell}{\partial W_{ij}} = \frac{\partial \ell}{\partial x_j} \cdot x_i \]

since

\[ \frac{\partial x_j}{\partial W_{ij}} = x_i \]
Gradient descent

\[ W_{ij}^{t+1} \leftarrow W_{ij}^t - \eta \cdot \frac{\partial \ell}{\partial W_{ij}} \]

by the chain rule

\[ \frac{\partial \ell}{\partial W_{ij}} = \delta_j \cdot x_i \]

also, define

\[ \delta_j := \frac{\partial \ell}{\partial x_j} \]
Weight updates

Gradient descent

\[ W_{ij}^{t+1} \leftarrow W_{ij}^t - \eta \cdot \frac{\partial \ell}{\partial W_{ij}} \]

by the chain rule

\[ \frac{\partial \ell}{\partial W_{ij}} = \delta_j \cdot x_i \]

How to compute?

\[ \delta_j := \frac{\partial \ell}{\partial x_j} \]
Backpropagation

Feedforward sweep

\[ x_{\text{out}} = \left( \prod_{i=1}^{i=#L} W_i \right) \cdot x_{\text{in}} \]

Backpropagation

\[ \delta^i = \left( \prod_{k=i+1}^{#L} W_k^T \right) \cdot \left( \nabla_{x_{\text{out}}} \ell \right) \]
Max-pooling

Pick element with max output in each block; zero out the rest.
During training, randomly zero out units with probability 0.5
ReLU networks

\[ \rho(x) = \max(0, x) \]
Path sums

\[
\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial x_j} \cdot \left( \frac{\partial x_j}{\partial x_i} \right)
\]

consider the internal structure
Path sums

\[
\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial x_j} \cdot \frac{\partial x_j}{\partial x_i}
\]

\[
\frac{\partial x_j}{\partial x_i} = \sum_{p \in \{i \to j\}} \prod_{\alpha \in p} w_\alpha
\]
Active path sums

\[ \frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial x_j} \cdot \frac{\partial x_j}{\partial x_i} \]

\[ \frac{\partial x_j}{\partial x_i} = \sum_{p \in \{i \rightarrow j\}_{active}} \prod_{\alpha \in p} w_{\alpha} \]
Knowledge distribution

- Units in NN $\leftrightarrow$ Players in a game
- All units have the same loss
- Loss is convex function of units’ weights when unit is active
- Units in output layer compute error-gradients
- Other units distribute signals of the form $(\text{error-gradients}) \times (\text{path-sums})$
Circadian games

- Units have convex losses when active
- Neural networks are circadian games
  - i.e. game that is convex for active (awake) players

**Main Theorem.** Convergence of neural network to correlated equilibrium is upper-bounded by convergence rates of individual units.

**Corollary.** Units perform convex optimization when active. Convex methods and guarantees (**on convergence rates**) apply to neural networks.

→ Explains why ReLUs work so well.
Philosophical Corollary

- Neural networks are games
  Units are no-regret learners

- Unlike many games (e.g. in micro-economic theory), the equilibria in NNs are extremely desirable

- Error-backpropagation distributes gradient-information across units

- **Claim:** There is a theory of distributed reasoning implicit in gradient descent and the chain rule.
The End. Questions?