Playing Peg Solitaire with Mathematical AI

Lecture 1 (G.H.): AlexNet and the modern AI toolkit — entropy, cross-entropy, gradient descent, backpropagation

Gerald Höhn (Kansas State University) and Philipp Höhn (Universität Bonn)

September 18, 2025

Overview

Lecture 1: AlexNet and the modern AI toolkit —
entropy, cross-entropy, gradient descent, backpropagation.

Lecture 2: Reinforcement learning —
Bellman equations, Q-learning, experience replay, and self-play on Peg solitaire.

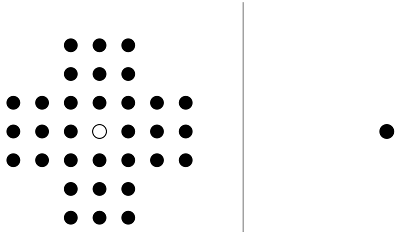
Lecture 3: Large language models —
from Transformers to reasoning models; vibe-coding a solver.

Prerequisites: Calculus, linear algebra; basic probability helpful.

Slides: `http://www.monstrous-moonshine.de/~gerald/talks/AI-lecture1.pdf`

Peg Solitaire — rules and plan

- Board: English 7×7 *cross* (33 holes).
- Move: jump an adjacent peg horizontally/vertically over a neighbor into an empty hole; the jumped peg is removed.
- Standard start: all holes filled except the center empty.
- Goal: end with a **single** peg in the center.
- Michael Collinson: ▸ Solving Peg Solitaire (2 minutes)



Start Goal

Prompt:

```
I like that you vibe code a trivial solver for peg solitaire for the
english board: There are 2^33  possible board positions. Looking at
the position with one peg in the middle we list all positions with
k pegs which have a valid move to a winning k-1 peg position. We also
list the move at each point. Checking for the initial 32 peg position
if it is in the winning set solves the game. Please write a short
python program implementing this simple alogithm
(note: I have a PC with 128 GB RAM).
```

251 lines of solver.py (generated by GPT-5)

```
# peg_retrograde.py
# Retrograde Peg Solitaire solver for the English board (33 holes).
# Finds whether the standard start (center empty) is in the winning set
# generated backwards from the center-one-peg goal. Optional path
    reconstruction.

from collections import defaultdict
from typing import List, Tuple, Dict, Set
import argparse

# ----- Board geometry (English 7x7 cross) -----
# Valid holes: positions (r,c) with r,c in 0..6 and (r in 2..4 or c in
    2..4)
# Map each valid (r,c) to a bit index 0..32
valid = []
idx_of = {}
for r in range(7):
```

[...]

Board coordinates (r, c) used by the solver

rows $r = 0 \dots 6$ (top to bottom)

		(0, 2)	(0, 3)	(0, 4)		
		(1, 2)	(1, 3)	(1, 4)		
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(2, 6)
(3, 0)	(3, 1)	(3, 2)	(3, 3) center	(3, 4)	(3, 5)	(3, 6)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(4, 6)
		(5, 2)	(5, 3)	(5, 4)		
		(6, 2)	(6, 3)	(6, 4)		

columns $c = 0 \dots 6$

Solver output: layers $k = 1..22$ with number of solvable board positions

```
k= 1  |layer|=1  (goal)
k= 2  |layer|=4
k= 3  |layer|=12
k= 4  |layer|=60
k= 5  |layer|=296
k= 6  |layer|=1338
k= 7  |layer|=5648
k= 8  |layer|=21842
k= 9  |layer|=77559
k=10  |layer|=249690
k=11  |layer|=717788
k=12  |layer|=1834379
k=13  |layer|=4138302
k=14  |layer|=8171208
k=15  |layer|=14020166
k=16  |layer|=20773236
k=17  |layer|=26482824
k=18  |layer|=28994876
k=19  |layer|=27286330
k=20  |layer|=22106348
k=21  |layer|=15425572
k=22  |layer|=9274496
k=23  |layer|=4792664
k=24  |layer|=2120101
k=25  |layer|=800152
k=26  |layer|=255544
k=27  |layer|=68236
k=28  |layer|=14727
k=29  |layer|=2529
k=30  |layer|=334
k=31  |layer|=32
k=32  |layer|=5
```

Solver output: one forward solution

```
Move  1: (5,3) jump over (4,3) to (3,3)
Move  2: (2,3) jump over (3,3) to (4,3)
Move  3: (3,5) jump over (3,4) to (3,3)
Move  4: (2,5) jump over (2,4) to (2,3)
Move  5: (3,2) jump over (3,3) to (3,4)
Move  6: (3,0) jump over (3,1) to (3,2)
Move  7: (3,3) jump over (3,2) to (3,1)
Move  8: (4,5) jump over (3,5) to (2,5)
Move  9: (4,3) jump over (3,3) to (2,3)
Move 10: (5,3) jump over (4,3) to (3,3)
Move 11: (4,0) jump over (4,1) to (4,2)
Move 12: (3,3) jump over (4,3) to (5,3)
Move 13: (2,0) jump over (3,0) to (4,0)
Move 14: (4,2) jump over (4,3) to (4,4)
Move 15: (4,4) jump over (3,4) to (2,4)
Move 16: (1,2) jump over (2,2) to (3,2)
Move 17: (2,4) jump over (2,3) to (2,2)
Move 18: (2,2) jump over (3,2) to (4,2)
Move 19: (3,3) jump over (3,2) to (3,1)
Move 20: (2,5) jump over (2,4) to (2,3)
Move 21: (6,2) jump over (5,2) to (4,2)
Move 22: (6,4) jump over (6,3) to (6,2)
Move 23: (3,0) jump over (3,1) to (3,2)
Move 24: (3,2) jump over (4,2) to (5,2)
Move 25: (1,2) jump over (2,2) to (3,2)
Move 26: (6,2) jump over (5,2) to (4,2)
Move 27: (2,0) jump over (2,1) to (2,2)
Move 28: (2,3) jump over (2,2) to (2,1)
Move 29: (4,2) jump over (3,2) to (2,2)
Move 30: (2,1) jump over (2,2) to (2,3)
Move 31: (1,3) jump over (2,3) to (3,3)
```

Mathematics as seen by an Administrator

Creating the Training Data

Mathematics as seen by an Administrator

Creating the Training Data

Input	→	Noisy Channel	→	Output
digits chosen from 1, 2, 3		Professor lecturing at blackboard		scribbled drawings for 1, 2, 3

Creating the Training Data

Input → Noisy Channel → Output

Input	Noisy Channel	Output
digits chosen from 1, 2, 3	Professor lecturing at blackboard	scribbled drawings for 1, 2, 3

Training the Classifier

Input → Classifier → Output

Creating the Training Data

Input → Noisy Channel → Output

Input	Noisy Channel	Output
digits chosen from 1, 2, 3	Professor lecturing at blackboard	scribbled drawings for 1, 2, 3

Training the Classifier

Input → Classifier → Output

Input	Classifier	Output
scribbled drawings of 1, 2 and 3	student trying to understand lecture	best guess from $\{1,2,3\}$

Mathematics as seen by a Statistician

Mathematics as seen by a Statistician

$$\boxed{\text{Input}} \longrightarrow \boxed{\text{Noisy Channel}} \longrightarrow \boxed{\text{Output}}$$

Random variable $Y \in \{1, 2, 3\}$;
$P(Y = i) = p_i, \ \ i = 1, 2, 3$
$\sum_{i=1}^{3} p_i = 1$

probability distributions $p(x \mid i)$
of scribbled digits
$x \in \mathbb{R}^{28\times 28}$ for each i

Random variable $X \in \mathbb{R}^{28\times 28}$
with probability distribution
$p_X(x) = \sum_{i=1}^{3} p_i \cdot p(x \mid i)$

Mathematics as seen by a Statistician

$$\boxed{\text{Input}} \longrightarrow \boxed{\text{Noisy Channel}} \longrightarrow \boxed{\text{Output}}$$

Input	Noisy Channel	Output
Random variable $Y \in \{1,2,3\}$; $P(Y = i) = p_i,\ \ i = 1,\,2,\,3$ $\sum_{i=1}^{3} p_i = 1$	probability distributions $p(x \mid i)$ of scribbled digits $x \in \mathbb{R}^{28\times 28}$ for each i	Random variable $X \in \mathbb{R}^{28\times 28}$ with probability distribution $p_X(x) = \sum_{i=1}^{3} p_i \cdot p(x \mid i)$

Taken input and channel together we have a random variable $Y \times X$ with probability distribution $p : \{1,2,3\} \times \mathbb{R}^{28\times 28} \longrightarrow \mathbb{R}$, $(i,x) \mapsto p(x,i)$. One has $p(x \mid i) = p(i,x)/p_i$.

Mathematics as seen by a Statistician

Random variable $Y \in \{1, 2, 3\}$;
$P(Y = i) = p_i, \ \ i = 1,\ 2,\ 3$
$\sum_{i=1}^{3} p_i = 1$

probability distributions $p(x \mid i)$
of scribbled digits
$x \in \mathbb{R}^{28\times 28}$ for each i

Random variable $X \in \mathbb{R}^{28\times 28}$
with probability distribution
$p_X(x) = \sum_{i=1}^{3} p_i \cdot p(x \mid i)$

Taken input and channel together we have a random variable $Y \times X$ with probability distribution $p : \{1, 2, 3\} \times \mathbb{R}^{28\times 28} \longrightarrow \mathbb{R}$, $(i, x) \mapsto p(x, i)$. One has $p(x \mid i) = p(i, x)/p_i$.

The distribution p has to be inferred by the prior knowledge $(i_j, x_j)_{j=1,\ldots,N}$ where $N \sim 10^5$–10^7.

Mathematics as seen by a Statistician

$$\boxed{\text{Input}} \longrightarrow \boxed{\text{Noisy Channel}} \longrightarrow \boxed{\text{Output}}$$

Random variable $Y \in \{1,2,3\}$;
$P(Y=i) = p_i, \quad i = 1,\,2,\,3$
$\sum_{i=1}^{3} p_i \;=\; 1$

probability distributions $p(x \mid i)$
of scribbled digits
$x \in \mathbb{R}^{28\times 28}$ for each i

Random variable $X \in \mathbb{R}^{28\times 28}$
with probability distribution
$p_X(x) = \sum_{i=1}^{3} p_i \cdot p(x \mid i)$

Taken input and channel together we have a random variable $Y \times X$ with probability distribution $p : \{1,2,3\} \times \mathbb{R}^{28\times 28} \longrightarrow \mathbb{R}$, $(i,x) \mapsto p(x,i)$. One has $p(x \mid i) = p(i,x)/p_i$.

The distribution p has to be inferred by the prior knowledge $(i_j, x_j)_{j=1,\ldots,N}$ where $N \sim 10^5$–10^7.

Then the classifier is the map (**Bayes classifier**)

$$c : \mathbb{R}^{28\times 28} \longrightarrow \{1,2,3\}, \qquad c(x) = \arg\max_{i\in\{1,2,3\}} p(i \mid x),$$

sending a picture x to the most likely digit i, i.e. the i for which $p(i \mid x) = p(i,x)/p(x)$ is maximal.

Solutions to the classification problem

- **Ill-posed** without any further assumptions on distribution $p(x, y)$.
- **Classical machine learning:** One may, for example, assume normal distributions.
- **Modern deep learning:** Cut $\mathbb{R}^{784}$ into 3 pieces by continuous piecewise affine functions realised by a deep neural network.
- **Latest method:** Upload picture of blackboard to a vision model. AI ignores prior knowledge $\{(x_i, y_i)\}_{i=1,\dots,N}$ and scribbled digits x completely. Instead realizes that the lecture is about Peano axioms for natural numbers and correctly reconstructs what the Professor *should* had written on the board.

Class-conditional model (class dependent Gaussians)

$$p(x \mid i) \sim \mathcal{N}(\mu_i, \Sigma_i), \qquad i \in \{1, 2, 3\}, \qquad p_i > 0, \ \sum_i p_i = 1.$$

Maximum-likelihood estimates from N_i samples per class

$$\hat{\mu}_i = \frac{1}{N_i} \sum_{k:y_k=i} x_k, \qquad \hat{\Sigma}_i = \frac{1}{N_i} \sum_{k:y_k=i} (x_k - \hat{\mu}_i)(x_k - \hat{\mu}_i)^\top, \qquad \hat{p}_i = \frac{N_i}{\sum_j N_j}.$$

Classical Machine Learning: Projecting to two dimensions via a generalized eigenproblem

Scatter matrices

$$\mu = \sum_{i=1}^{3} p_i \mu_i, \qquad S_B = \sum_{i=1}^{3} p_i (\mu_i - \mu)(\mu_i - \mu)^\top, \qquad S_W = \sum_{i=1}^{3} p_i \Sigma_i.$$

- With 3 classes, $\operatorname{rank}(S_B) \leq 2$.
- Find plane $W = [v_1 \ \ v_2]$ in $\mathbb{R}^{784}$ from

$$S_B v = \lambda S_W v, \ \text{take the eigenvectors for the two top eigenvalues of } S_W^{-1} S_B$$

Interpretation: W maximizes $J(W) = \operatorname{tr}\big((W^\top S_W W)^{-1} W^\top S_B W\big)$, i.e. between-class mean separation relative to within-class scatter.

Classical Machine Learning: Bayes estimates after projection to the plane

Projected parameters in $\mathbb{R}^2$

$$x' = W^\top x \in \mathbb{R}^2, \qquad \mu_i' = W^\top \mu_i \in \mathbb{R}^2, \qquad \Sigma_i' = W^\top \Sigma_i W \in \mathbb{R}^{2\times 2}.$$

Bayes estimates in $\mathbb{R}^2$

$$g_i'(x') = -\tfrac{1}{2}(x' - \mu_i')^\top (\Sigma_i')^{-1}(x' - \mu_i') - \tfrac{1}{2}\log\det\Sigma_i' + \log p_i, \qquad \hat{\imath}(x') = \arg\max_i g_i'(x').$$

Definition (Affine Map)

A function $L:\mathbb{R}^n\longrightarrow\mathbb{R}^m$, $x\mapsto Ax+b$, where $A\in\mathrm{Mat}(m\times n,\mathbb{R})$ and $b\in\mathbb{R}^m$.

Definition (ReLU (Rectified Linear Unit) function)

The function $\rho:\mathbb{R}\longrightarrow\mathbb{R}$, $x\mapsto\rho(x)=\mathrm{ReLU}(x)=\max(0,x)=\begin{cases}0, & x<0,\\ x, & x\geq 0,\end{cases}$ extended componentwise to $\rho:\mathbb{R}^n\longrightarrow\mathbb{R}^n$.

Definition (Deep neural network with $\ell-1$ hidden layers))

Fix widths $d_0=n$, d_1, ..., $d_\ell=m$. Let $L_i(x)=A_ix+b_i$ with $A_i\in\mathrm{Mat}(d_i\times d_{i-1})$, $b_i\in\mathbb{R}^{d_i}$. A **deep neural network** is a continuous piecewise affine function $n_\theta:\mathbb{R}^n\longrightarrow\mathbb{R}^m$ of the form

$$n_\theta=L_\ell\circ\rho\circ L_{\ell-1}\circ\cdots\circ\rho\circ L_2\circ\rho\circ L_1.$$

One calls $\theta=(A_1,b_1,A_2,b_2,\ldots,A_\ell,b_\ell)$ the **parameters** of the net.

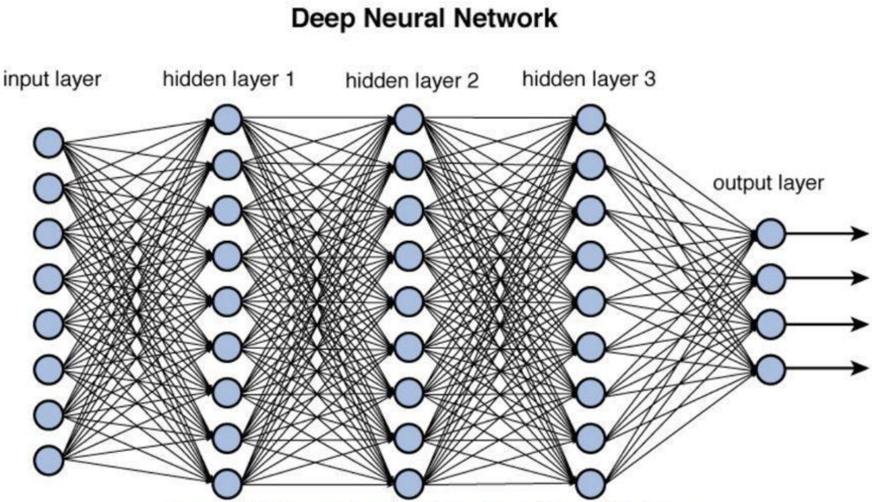


Figure 12.2 Deep network architecture with multiple layers.

Deep Neural Networks: The Training

Deep Neural Networks: Entropy and Conditional Entropy

Physics Origin (Thermodynamic, Statistical Physics)
19th century: Carnot, Clausius, Gibbs, Boltzmann

Information Theory (Shannon 1948): *A Mathematical Theory of Communication*

For a random variable Y with n possible outcomes with probablities $p_1, \ldots p_n$ the **entropy** is

$$H_n(p_1, \ldots, p_n) = \sum_{i=1}^{n} -p_i \cdot \log_2 p_i .$$

measuring the uncertainty about the outcome y of Y in **bits**.
The $\{H_n\}_{n=1,2,\ldots}$ are the unique (up to a positive constant) system of continuous functions satisfying natural axioms about information.

After sending digit y through our noisy channel and watching x, the uncertainty about which digit was chosen is reduced. The remaining uncertainty is measured by the **conditional entropy**

$$H_p(Y|X) = -\sum_{y\in\{1,2,3\}} \int_{x\in\mathbb{R}^{784}} p(x,y) \cdot \log p(y|x)\, dx .$$

Let $\mathrm{softmax} : \mathbb{R}^n \longrightarrow \Delta^{n-1} \subset \mathbb{R}^n$, $x \mapsto (e^{x_i} / \sum_{j=1}^n e^{x_j})_{i=1,\ldots,n}$ and set $q_\theta = \mathrm{softmax} \circ n_\theta$.

Training our net means finding θ such that $q_\theta(x)$ becomes a good approximation for $p(y|x)$, so that knowing x we can predict/select the most likely digit y.

We define the **loss function** $\mathcal{L}(\theta)$ as the **cross entropy**

$$\mathcal{L}(\theta) = - \sum_{y \in \{1,2,3\}} \int_{x \in \mathbb{R}^{784}} p(x, y) \log q_\theta(y|x)\, dx.$$

One has

$$\mathcal{L}(\theta) = H_p(Y|X) + \int_{x \in \mathbb{R}^{784}} \Big(\sum_{y \in \{1,2,3\}} p(x, y) \Big) \cdot KL(p(\,.\,|x) || q_\theta(\,.\,|x))\, dx \geq H_p(Y|X).$$

To minimize $\mathcal{L}(\theta)$, or more precisely its estimate $\widehat{\mathcal{L}}(\theta) = -\frac{1}{N} \sum_{i=1}^N \log q_\theta(y_i|x_i)$, we use stochastic **gradient descent** $\theta_{t+1} = \theta_t - \alpha \nabla_\theta \widehat{\mathcal{L}}(\theta_t)$ by estimating $\nabla_\theta \widehat{\mathcal{L}}(\theta_t)$ through averaging over small samples (batches) B_t of the training data set $(x_i, y_i)_{i=1,\ldots,N}$. Gradients are computed via **backpropagation**. The parameter α is called the **learning rate**.

Videos on backpropagation / gradient descent and on AlexNet

3Blue1Brown: ▸ Backpropagation and gradient descent (10 minutes)

Welch Labs: ▸ AlexNet (5 minutes)