REINFORCEMENT LEARNING

Introduction
Basic principles

Diego Klabjan
Professor, Industrial Engineering and Management Sciences

Northwestern | McCORMICK SCHOOL OF ENGINEERING

Credit

- Several slides adaption of the lecture material from
  - Sergey Levine from Berkeley
    - http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html
  - David Silver from University College London
    - http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
- A few examples taken from Lex Fridman, MIT
  - https://selfdrivingcars.mit.edu
- Tremendous acknowledgment to Hindeke Tewodros
  - Prepared all of the slides
  - Undergraduate student at Northwestern

Outline

- Basic principles
- Components
  - State
  - Action
  - Policy
  - Environment
- Review of Markov Chains
- Review of gradient optimization
- Applications
  - Games
  - Chess/AlphaGo
  - Business applications
    - Manufacturing
    - Energy
    - Supply chain management

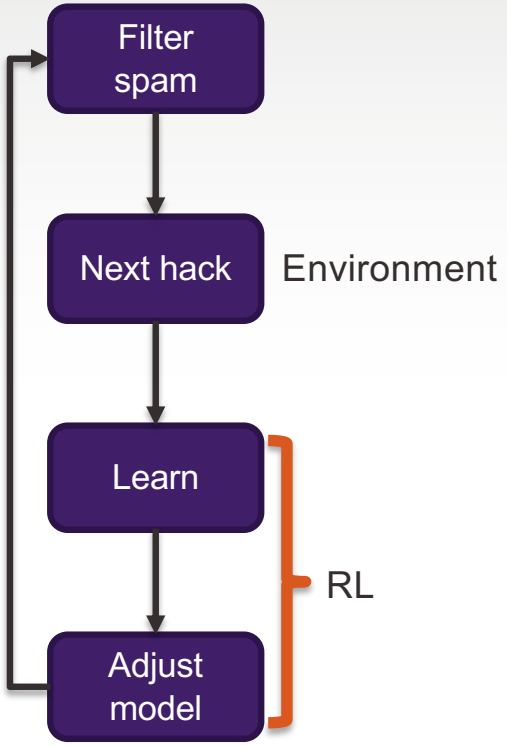
BASIC CONCEPTS

Supervised, Unsupervised

- Data consisting of feature vector and optional labels
- Unsupervised
  - No labels
  - Customer segmentation
    - Any clustering
- Supervised
  - Predict labels to unseen data
  - Pricing for logistic services
  - Predict demand for bikes
    - Bike-sharing program
- No feedback loop
  - Email spam filter counterattacked my spammers

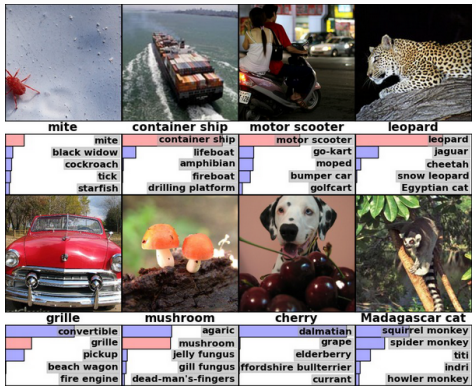
Reinforcement Learning

- Algorithm that learns on its own
- Spam filter
  - New hack
  - Algorithm learns it
  - Automatically adjust strategy
- Supervision/labels replaced by reward
  - Make adjustment and model rewarded for it
  - Adjustment
    - Purchase extra 10 units of stock due to cold weather
    - Reward is revenue for these units – procurement cost

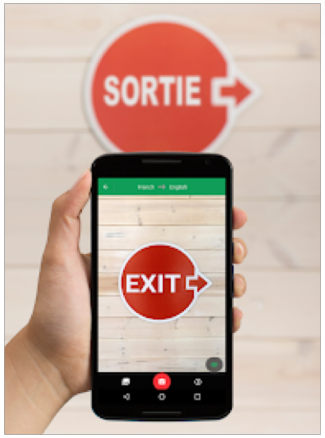


One-off Decision Making

- When system making a single isolated decision
  - Classification, regression
- When that decision does not affect future decisions



http://rocknrollnerd.github.io/ml/2015/05/27/leopard-sofa.html



https://play.google.com/store/apps/details?id=com.google.android.apps.translate&hl=en

Sequential Decision Making

- Limited supervision
  - You know **what** you want, but not **how** to get it
- Actions have consequences
  - Future reward unknown

Caveat: Which ones have been operationalized?

Notorious applications

robotics

http://blogs.wiscanada.com/content/will-social-services-be-changed-by-artificial-intelligence-and-robotics/

autonomous driving

https://www.aspeninstitute.org/tag/autonomous-vehicles/

language & dialogue (structured prediction)

https://www.dreamstime.com/stock-photo-people-talk-3d-social-media-speech-bubbles-image16446490

operations management

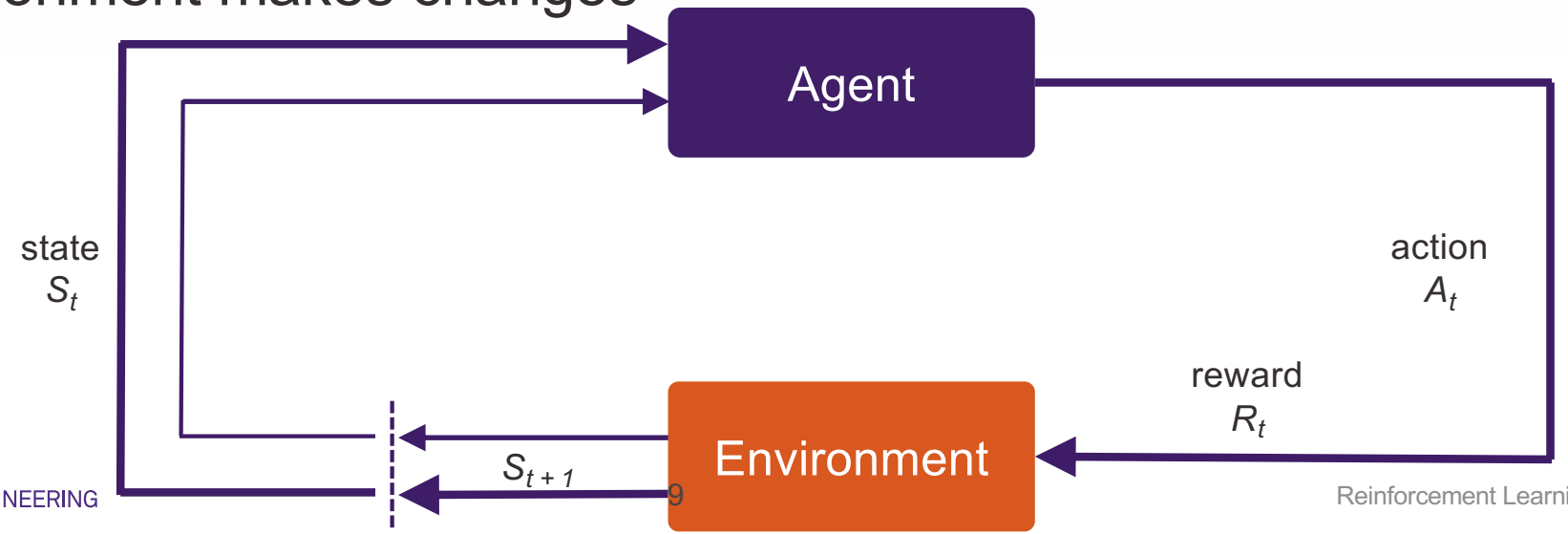
https://www.mainline.com/services/assess/

finance

https://www.123rf.com/photo_13129528_business-background-with-finance-graphs-pen-and-calculator.html

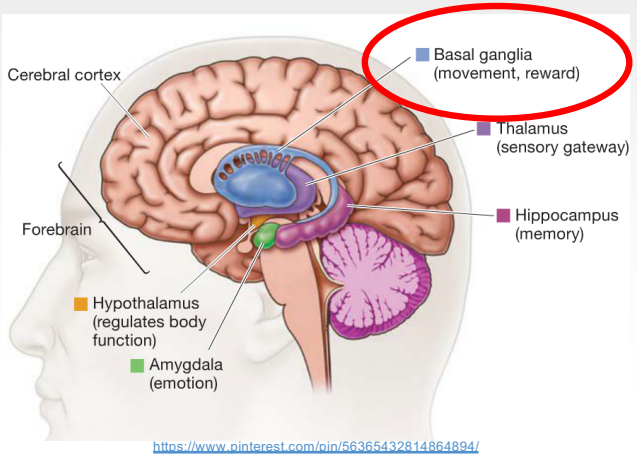
Setting and Model

- State
- Action
- Get reward
- Environment makes changes



Reward

http://funkypotato.com/pac-man/



https://www.pinterest.com/pin/563654328148648 94/

https://steemit.com/science/@coinbizpro/who-are-you-a-predator-or-a-victim

Learning Types

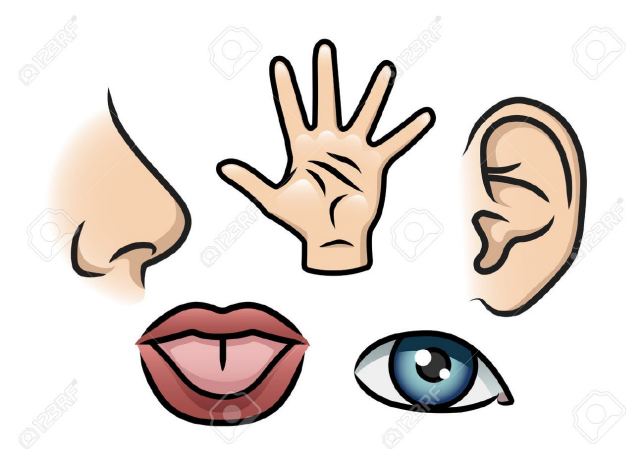
- Learning from demonstrations
  - Directly copying observed behavior
  - Infer reward
  - Imitation learning
- Learning from observing the world
  - Unsupervised learning
- Learning from other tasks
  - Transfer learning
  - Meta-learning: learning to learn

Imitation Learning

https://www.motorauthority.com/news/1031363_volvo-launches-research-project-to-map-driver-behavior

From Input to Output

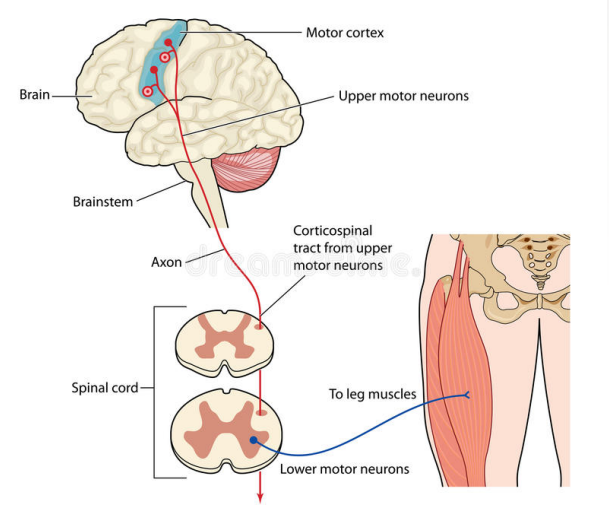
Sensory inputs



https://www.123rf.com/photo_26573894_stock-vector-a-cartoon-illustration-depicting-the-5-senses-smell-touch-hearing-taste-and-sight.html

DNN to encode state

Output: complex actions



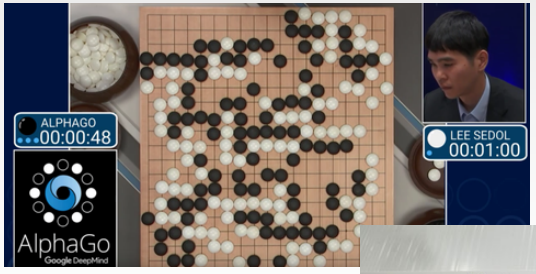
https://www.dreamstime.com/stock-illustration-motor-nerves-leg-to-motor-cortex-originating-muscles-traveling-via-spinal-cord-brain-created-adobe-image61090743

DNN model actions

Capability Today

- Acquire high degree of proficiency in domains governed by simple, known rules
- Learn simple skills with raw sensory inputs
- Learn from imitating enough human-provided expert behavior

https://www.popularmechanics.com/technology/a19863/google-alphago-ai-wins-second-game-go/



https://spectrum.ieee.org/automaton/robotics/humanoids/space-justin

https://blog.carsforsale.com/top-10-affordable-cars-with-a-backup-camera/

Limitations Today

- Humans can learn incredibly quickly
  - Deep RL methods are usually slow
- Humans can reuse past knowledge
  - Transfer learning in deep RL is an open problem
  - AlphaGo cannot play Atari
    - Even less optimize inventory
- Reward challenging
  - Read Super Intelligence by Nick Bolstrom

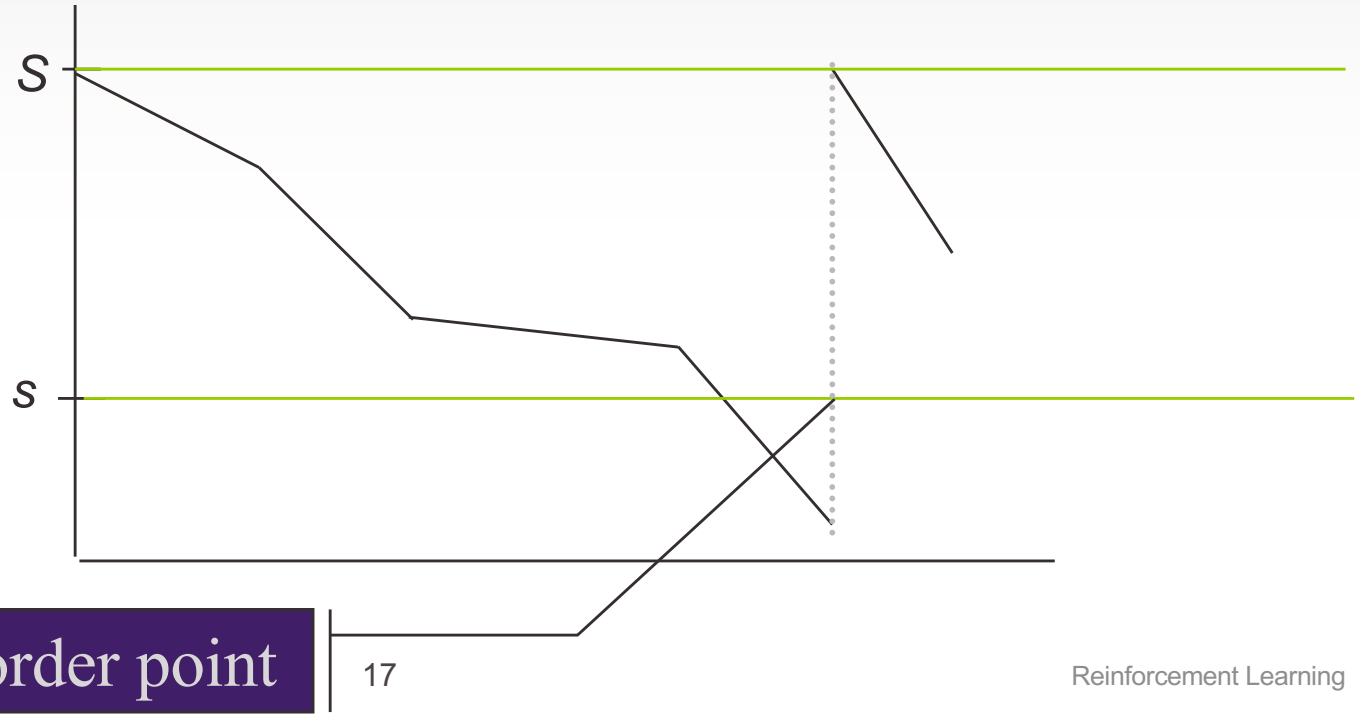
Inventory Management

- Retail store with SKU's
  - Brick-and-mortal or e-commerce
- Have 10 pairs of Nike Air Jordan
- Should we order additional boxes?
  - Forecast to sell 5 tomorrow
  - Options
    - Do not order
    - Order 5 units
    - Order 50 units

www.niketalk.com

(s,S) Policy

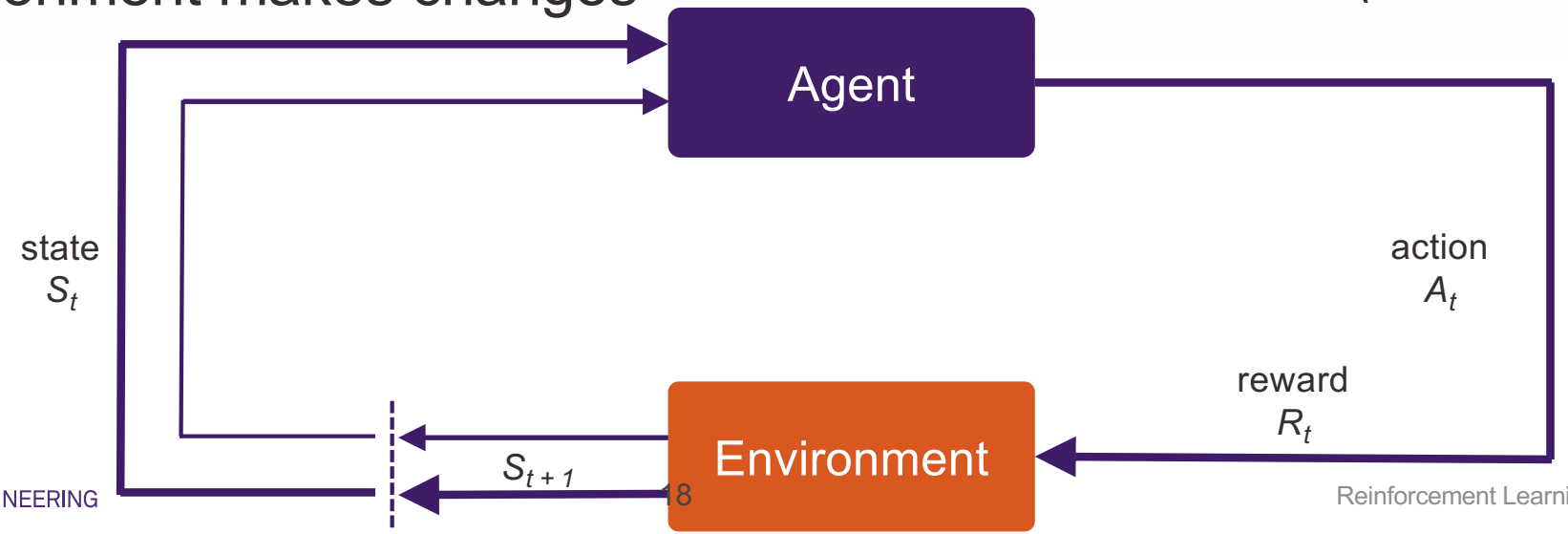
- Demand is stochastic
- Policy to order when inventory below s
  - Order up to S



Setting and Model

- State
- Action
- Get reward
- Environment makes changes

- *Inventory level*
- *How much to order*
- *Reward of revenue - cost*
- *Random demand (adversary)*



Inventory Management

- State = inventory units at 8 am
- Action = how many to order of each SKU
- Reward = selling price of units – cost of procurement
- Environment = stochastic demand

$$S_{t+1} = S_t + a_t - D_t$$

$$R(S_t, a_t) = p_t D_t - c_t a_t - U_t 1_{a_t > 0}$$

- What if demand exceeds inventory plus order?
- What if orders have lead time?

APPLICATIONS

Portfolio Management

- Cash and list of security to invest
  - Can change position every morning
- State
  - Number of positions in each security and cash
- Action
  - How much to buy/sell of each security
- Reward
  - Profit and loss
  - Reality much more complex

$$reward = utility(wealth)$$

- Environment
  - Interest rates on cash
  - Liquidity of securities

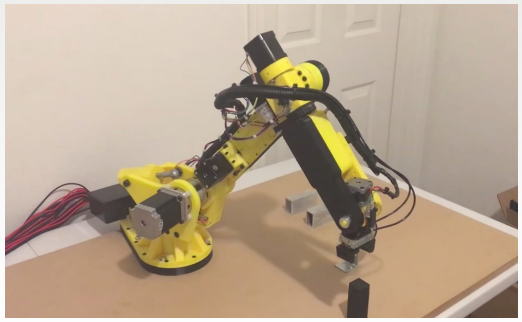
Tic-tac-toe

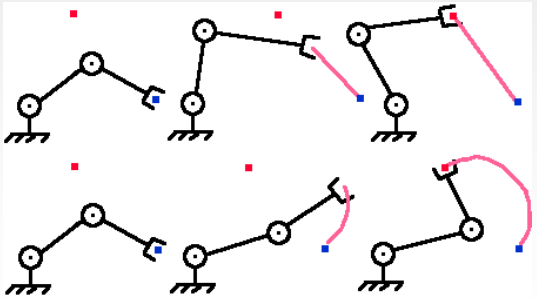
- State = positions on board
- Action = next move
- Reward
  - Zero if not end
  - 1 if win
  - -1 if we lose
- Environment
  - Move of opponent

	x	
x	o	
	o	

Robotics

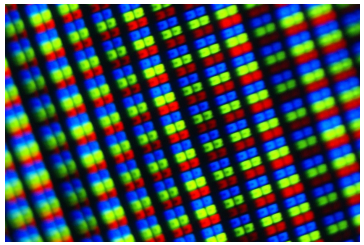
https://www.fabbaloo.com/blog/2017/7/3/design-of-the-week-six-axis-robot-arm

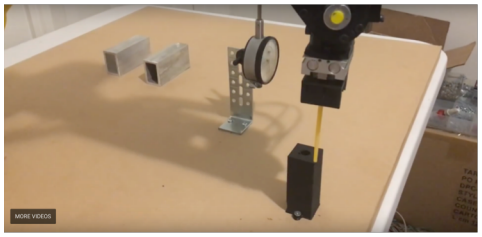


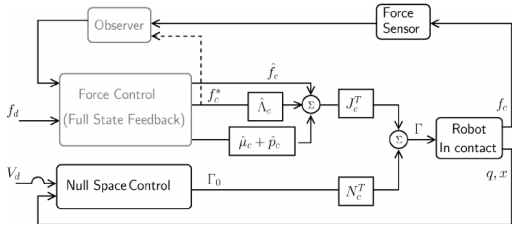


robotic control pipeline

observations → state estimation (e.g. vision) → modeling & prediction → planning → low-level control → controls



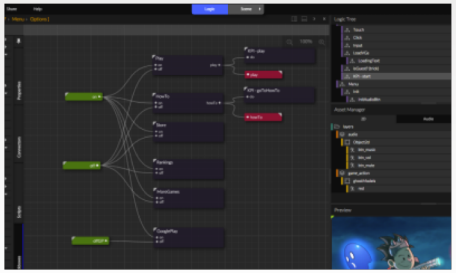




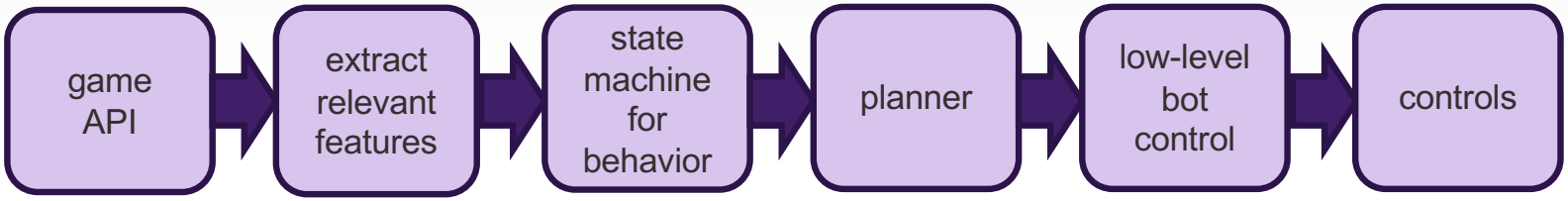
Playing Video Games

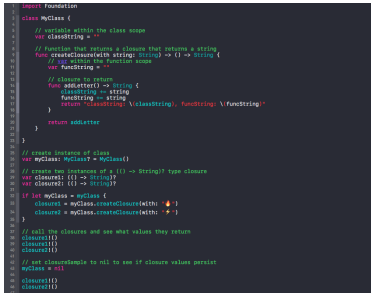
https://www.wired.com/2010/03/red-steel-2-review/

https://elearningindustry.com/directory/elearning-software/wimi5



video game AI pipeline

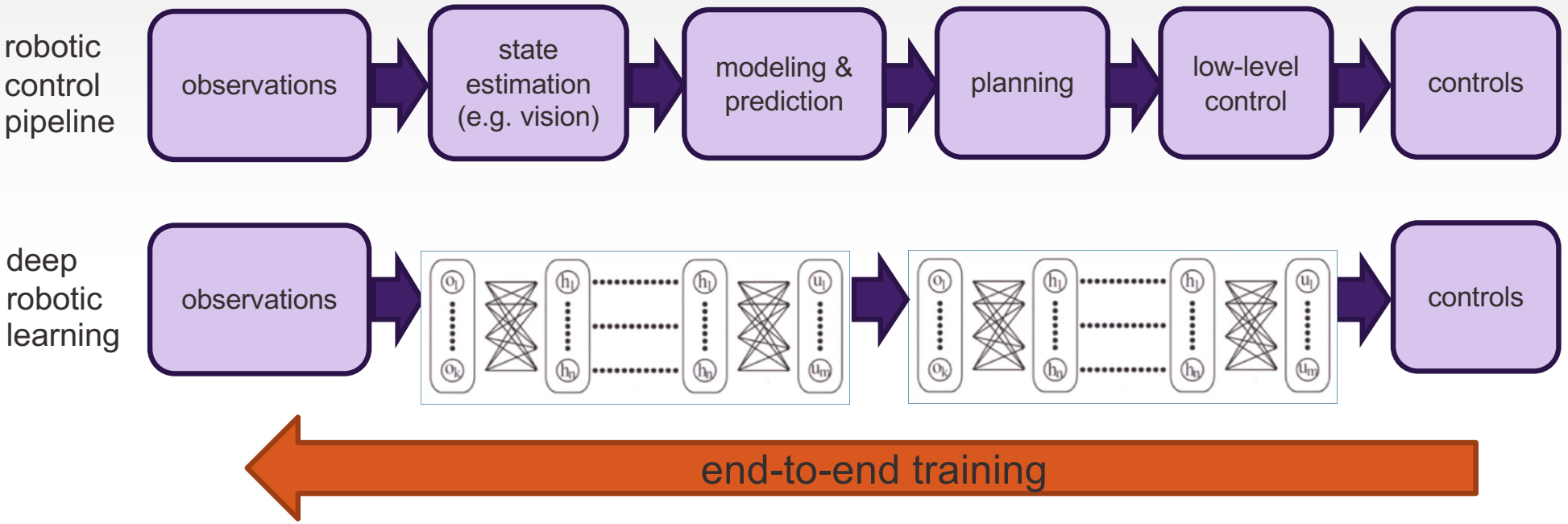




https://medium.com/@Dougly/closures-with-swift-58b274d849ad

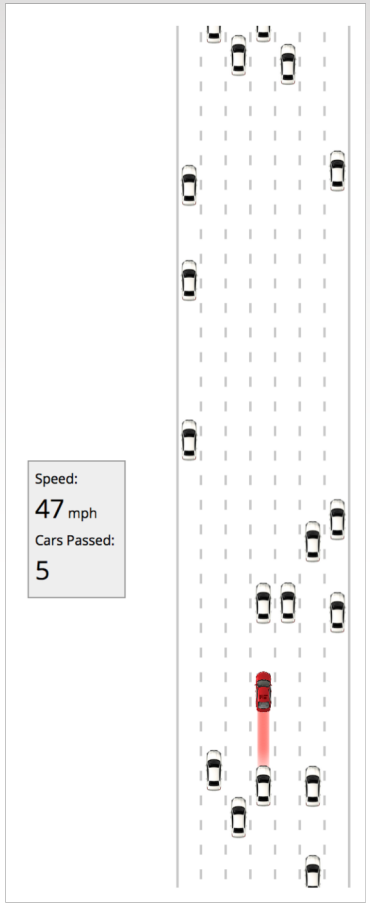
https://www.theverge.com/2014/12/16/7382735/the-best-video-games-of-2014

Deep Reinforcement Learning



https://selfdrivingcars.mit.edu/deeptraffic/

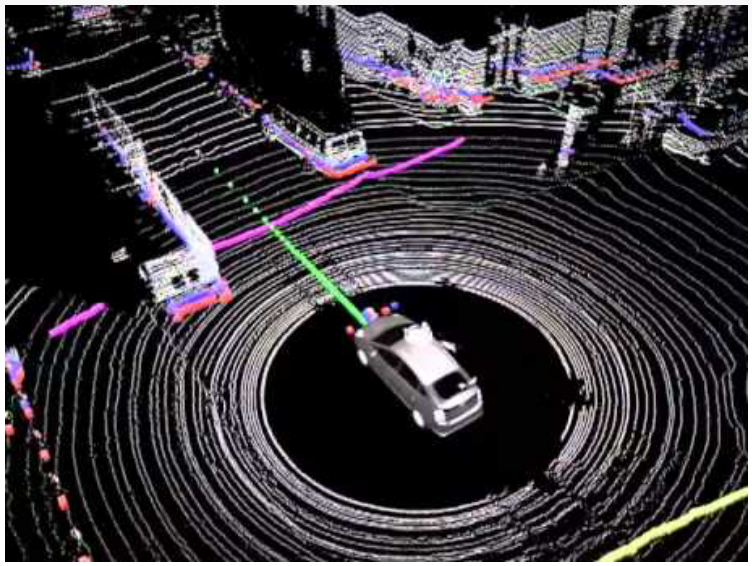
- Discrete time (1 second)
- State
  - Location of all cars
    - Deep learning encoding
    - Object recognition
  - Your own location
- Action
  - Position of your car next second
  - In cars translate to control
- Reward
  - Cost of crash, maintain speed, etc
- Environment
  - Obstacles on road, pedestrians, etc



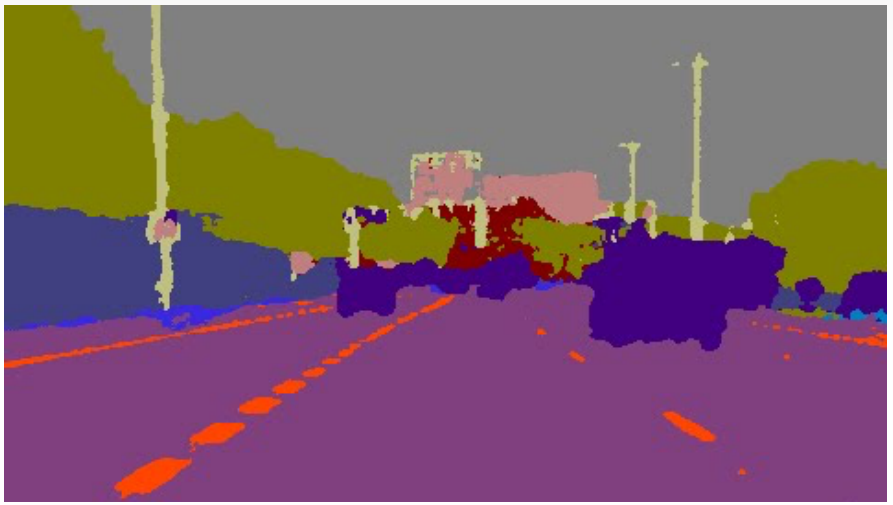
Autonomous Cars

Autonomous Cars

Lidar https://youtu.be/nXlqv_k4P8Q



Camera https://youtu.be/CxanE_W46ts



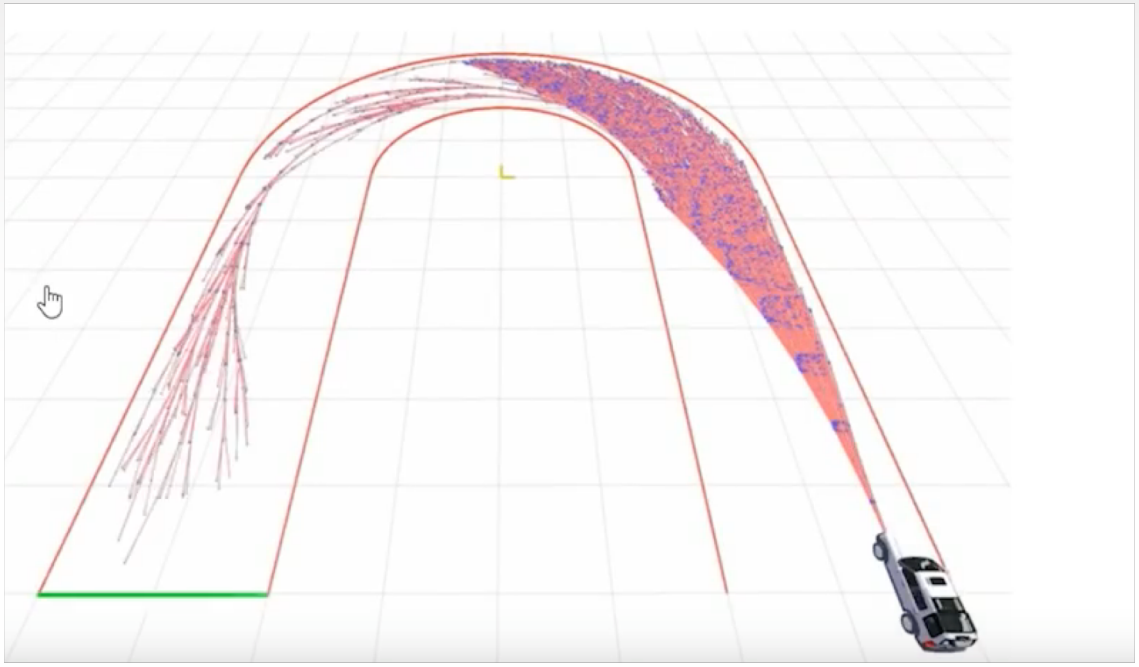
Autonomous Cars

- Scene understanding
  - Deep learning for bounding box object detection
- Two tasks solved by deep learning
  - Current position of the car
  - Surrounding bounding box with classification
- Traditional approach
  - Generate possible trajectories
  - Find the best one by optimization
- Modern is reinforcement learning

https://youtu.be/aKed5FHzDTw

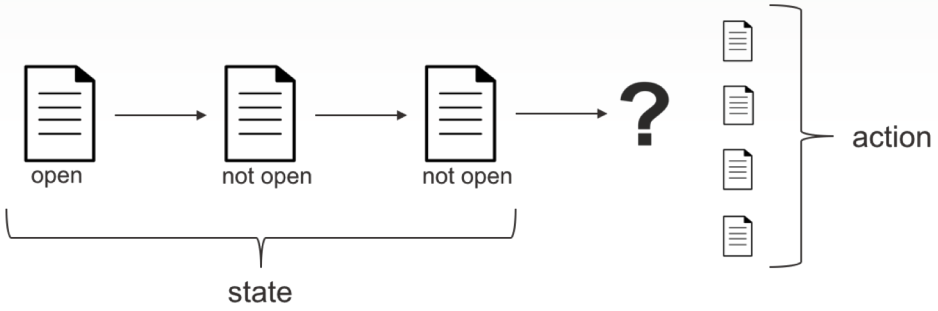
https://youtu.be/_LJefOSZ608

Autonomous Cars



Information Retrieval

- Problem
  - Query and set of retrieved documents
  - Order the documents based on relevance
- State
  - Documents already displayed
- Action
  - Next document to display
- Reward
  - Relevance score of document
    - Revealed only after document displayed
      - User opened



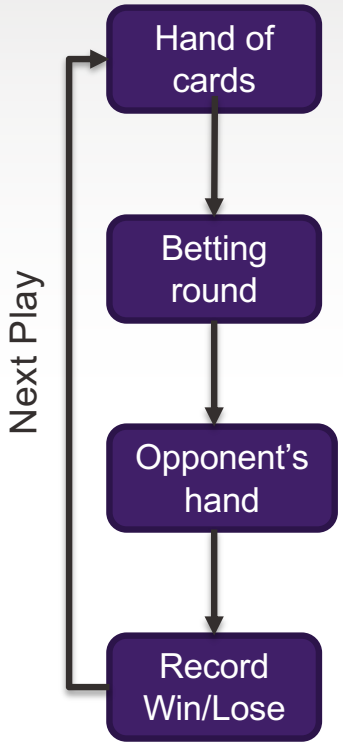
Airline Revenue Management

- Two classes: Basic economy and refundable ticket
  - Limited number of seats on aircraft
- Passenger places a booking request
- Action
  - Accept the request or reject
  - Why not always accept?
- State
  - Available seats in aircraft
- Reward
  - Price of itinerary

- Environment
  - Do not know future booking requests
- Dilemma
  - Economy request
  - If accept, a refundable booking request might be coming
  - If reject, seat might be left unfilled

Use Cases

- Production planning
  - Current state of a plant
  - Schedule production for next hour
  - Reward
  - Machines go down, electricity rates spike
- Chess (games in general)
  - State of the board
  - Next move
  - Leads to a win
  - Opponent makes the move
- Shortest path
- AlphaGo



Contextual Bandit

- User with profile access a web page
- Action
  - Ads to display
- Reward
  - Click on ad
  - Caveat: revealed only after action made
- State
  - Features of user and candidate ads
- Next state?

- Not reinforcement learning
  - Next state independent of action and current state
  - No notion of future reward based on current action

Power Bidding

- Generators bid for power generation during next 24 hours
  - Action: Each one submit quantity and price
- State
  - State of grid, load, generator
- Is it reinforcement learning
- No
  - State tomorrow does not matter of action
- What if long term requirement of minimum or maximum generated quantity?

Single Agent vs Multi Agent

- Single agent
  - Single decision maker
  - In games other players captured in environment
  - Reasonable when many participants
    - Individual behaviors neutralize
- Multi agent
  - Explicitly model competing agents
  - Use when powerful dominant competing agents
    - Large trading firm
  - Texas Hold'em
    - Imperfect information game

- Multi agent modeling
  - State captures all agents
  - Action for each individual agent
  - Reward for each agent
  - Next state depends on state and actions of each agent
- Agents can compete or collaborate

Summary

- Reinforcement learning: state, action, reward, environment
- Imitation learning
  - Reward not explicitly given, episodes are given
- Contextual bandit
  - State, action, reward revealed after action
  - No notion of next state
- Single agent vs multi agent

MARKOV CHAINS

Example

- Inventory in stock at beginning of day t, for t = 0,1,2,…,
  - X_t - these are the states
  - One possible sample path
    - X_0=3, X_1=0, X_2=4, X_3=1
  - What is the probability of this sample path?

$$P(X_3 = 1, X_2 = 4, X_1 = 0, X_0 = 3) = P(X_3 = 1 \mid X_2 = 4, X_1 = 0, X_0 = 3)$$
$$\times P(X_2 = 4 \mid X_1 = 0, X_0 = 3) \times P(X_1 = 0 \mid X_0 = 3) \times P(X_0 = 3)$$

Example

- The process gets pretty cumbersome!
- Under Markov property

$$P(X_3 = 1, X_2 = 4, X_1 = 0, X_0 = 3) = P(X_3 = 1 \mid X_2 = 4) \times P(X_2 = 4 \mid X_1 = 0) \times P(X_1 = 0 \mid X_0 = 3) \times P(X_0 = 3)$$

- The conditional probability of tomorrow's inventory level depends only on what we have today
  - Not on entire inventory history!

Required Problem Knowledge

- All possible states of a system
  - Status of the system at some point in time
- The probabilities of transitioning between states
  - Represent the probability of changing from a state to another state at some point in time.
    - Also called “one-step” transition probabilities
  - Stationary transition probabilities
    - Does not depend on time

$$p_{ij} = P\left(X_{t+1} = j \mid X_t = i\right) = P\left(X_1 = j \mid X_0 = i\right)$$

Required Problem Knowledge

- Initial condition
  - Unconditional probability of each state at the beginning of process
  - The probability that we start in state *i*

$$P(X_0 = i)$$

Properties of Markov Chains

- Stochastic process with the Markovian property

$$p_{ij} = P\left(X_{t+1} = j \mid X_t = i\right) = P\left(X_1 = j \mid X_0 = i\right)$$

transition probability

- History does not matter – except the last step

$$P\{X_{t+1} = j \mid X_0 = k_0, X_1 = k_1, \cdots, X_{t-1} = k_{t-1}, X_t = i\}$$
$$= P\{X_{t+1} = j \mid X_t = i\}$$

Example

- An employee either shows up for work or is absent.
  - If an employee shows up today and yesterday, with probability .9 she shows up tomorrow.
  - If an employee does not show up today nor yesterday, with probability .2 she shows tomorrow.
  - If an employee shows up today, but not yesterday, with probability .3 she shows up tomorrow.
  - If an employee showed up yesterday, but not today, with probability .8 she shows up tomorrow.

Example

- State = 0 if an employee shows up today and yesterday
- State = 1 if an employee shows up today but not yesterday
- State = 2 if an employee does not show up today, but did yesterday
- State = 3 if an employee does not show up either today or yesterday

Example

- Transition matrix P

$$\begin{pmatrix} .9 & 0 & .1 & 0 \\ .3 & 0 & .7 & 0 \\ 0 & .8 & 0 & .2 \\ 0 & .2 & 0 & .8 \end{pmatrix}$$

Main Result

- Suppose we start with an initial probabilistic vector x
- The probability that after n steps we are in state j is $\left(xP^n\right)_j$
  - The j-th coordinate in xP^n

Example

- If an employee shows up today but not yesterday (state 1)

$$x = (0,1,0,0)$$

- xP = (.3 0 .7 0) = distribution of what state the employee will be in on the following day
- $(xP)P = xP^2$ = (.27 .56 .03 .14) = distribution of what state the employee will be in two days later

Remarks

- $x = \underbrace{(0, 0, \cdots, 0, 1, 0, 0, \cdots 0)}_{i\text{th coordinate}}$
  - Start in state *i*
- Probability of being in state *j* after *n* steps

$$P_{ij}^{n}$$

- If *x* is a 'true probabilistic' vector
  - Start in a random state
  - Each initial state has its corresponding probability

Chapman-Kolmogorov Equations

- For any *k*

$$p_{ij}^{(n)} = \sum_{h=1}^{m} p_{ih}^{(k)} p_{hj}^{(n-k)}$$

- The probability of being in state *j* after *n* steps
  - Decomposed into the probability of being in state *h* after *k* steps
  - Then moving from state *h* to *j* in the remaining *n-k* steps
  - Provided we account for all possible intermediate states *h*

Steady-State Probabilities

- For some Markov chains
  - Long-run behavior is independent of how the process begins
- Recall pageRank

$$\pi_j = \lim_{n \to \infty} p_{ij}^{(n)}$$

- Keep walking for a long time and count at what state stopped

Steady-State

- Equations

$$\underbrace{\pi = \pi P}_{a\ vector-matrix\ product} \qquad \underbrace{\sum_{all\ j} \pi_j = 1}_{a\ vector\ norm}$$

- If we have 1,...,*M* states, then we have *M*+1 equations in *M* unknowns.
- π is a left eigenvector, with eigenvalue 1.
  - Also a probability distribution!
- Can use typical matrix methods to solve for π.
  - Gaussian elimination, decomposition, find the inverse, etc.

Interpretation

- Station probability exists only for certain Markov chains
  - Ergodic Markov chains (aperiodic and irreducible)
- If the process runs for a long time, then π_i is the probability of being in state *i*
  - The process is stationary at this point
- This probability does not depend on the initial state
- The percentage of time spent in state *i*

Expected Costs

- Interpret π as the long-run % of time the process in state *i*
- Long-run costs or revenues associated with each state
  - Average cost

$$Expected\ \text{cost}/unit\ time = \sum_{j=0}^{M} \pi_j C(j)$$

(sum over all costs, each weighted by its probability of occurrence)

Example

- Solving $\pi = \pi P$ leads to $\pi = (45, 21, 10) / 76$.

$$P = \begin{pmatrix} .7 & .1 & .2 \\ .5 & .5 & 0 \\ .3 & .6 & .1 \end{pmatrix}$$

- The linear equations to solve are

$\pi_1 = .7\ \pi_1 + .5\ \pi_2 + .3\ \pi_3$
$\pi_2 = .1\ \pi_1 + .5\ \pi_2 + .6\ \pi_3$
$\pi_3 = .2\ \pi_1 + 0\ \pi_2 + .1\ \pi_3$
$1 = \pi_1 + \pi_2 + \pi_3$
$\pi_1 \geq 0, \pi_2 \geq 0, \pi_3 \geq 0$

GRADIENT OPTIMIZATION

Loss Function

- $\sum_{x,y} l(x, y; w)$
  - x,y from train dataset
  - w model parameters
- Alternative viewpoint
  - Function that transforms x close to y
  - $f(x; w) \approx y$
- $\sum_{x,y} g(f(x; w), y) = \sum_{x,y} l(x, y; w)$
  - Function g captures penalty for not being perfect

$$-\ln \frac{1}{1+e^{-wxy}}$$

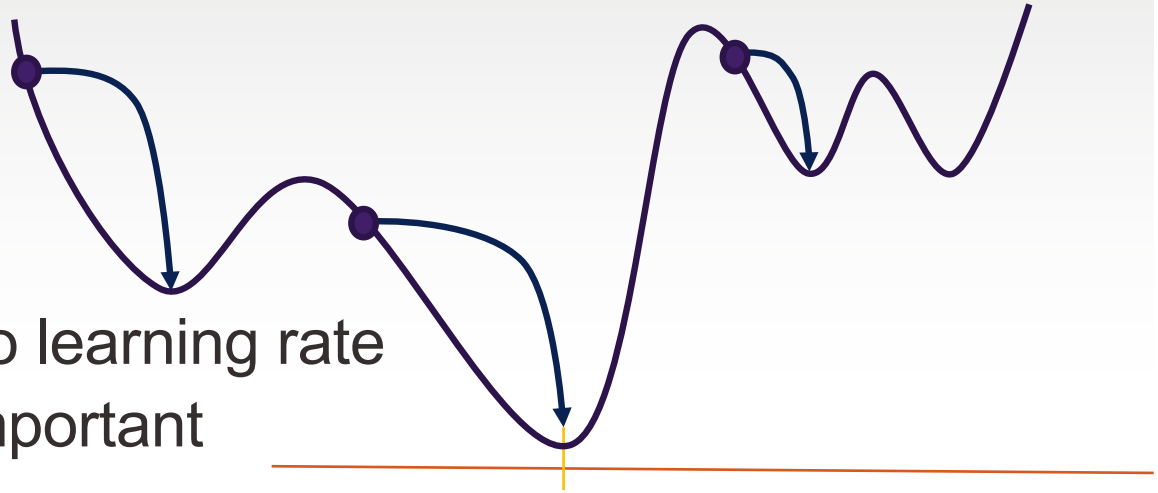
$$(y - g(x; w))^2$$

$$\max(0, 1 - wxy)$$

$$\min_w \sum_{xy} \ell(x, y; w)$$

Gradient Optimization

$$w_{m+1} = w_m - t_m \nabla f(w_m)$$



- Algorithm very sensitive to learning rate
- Starting point also very important
- Convergence
  - Certain learning rates lead to convergence
  - No guarantee to optimal solution
    - True only if function convex

Gradient

- Square error

$$\frac{\partial w}{\partial w_k}\sum_i (y_i - wx_i)^2 = -2\sum_i x_{ik}(y_i - wx_i)$$

- Logistic regression

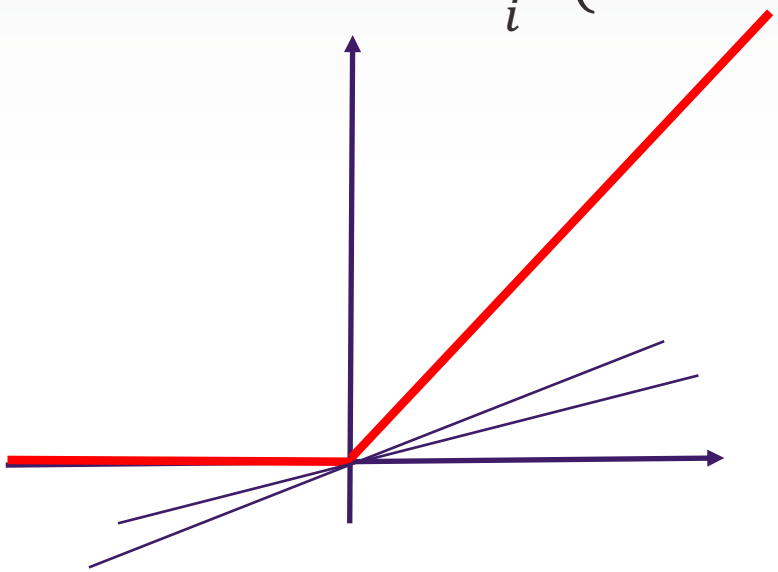
$$\frac{\partial w}{\partial w_k}\sum_i -ln(1+\exp(-y_i w x_i))^{-1}$$

$$= \sum_i -y_i x_{ik}\exp(-y_i w x_i)\,(1+\exp(-y_i w x_i))^{-1}$$

Gradient

- Hinge loss

$$\frac{\partial w}{\partial w_k}\sum_i \max(0,1-y_i w x_i) = \sum_i \begin{cases} -y_i x_{ik} & y_i w x_i < 1 \\ 0 & y_i w x_i \geq 1 \end{cases}$$



Gradient Optimization for ML

- Loss function involves all samples
  - Gradient is the sum of all gradients
  - Computationally intractable
- Solution
  - Randomly select subset S of samples
  - Compute the gradient for each selected sample
  - Average them
- Both solutions allow parallelization
  - Embarrassingly parallel

$$w_{m+1} = w_m - t_m \frac{1}{|s|} \sum_{i \in S} \nabla l(x_i, y_i; w_m)$$

Pros and Cons

Full gradient

High per iteration time
Low number of iterations
Most stable
Very low variance

Stochastic gradient descent

Lower per iteration time
Higher number of iterations
Harder to set learning rate
Higher variance

Adaptive: Adagrad

- Same learning rate for all weights
- Sparse feature (rare words in NLP) can be very important
  - Should get high learning rate
  - Likely to get small derivatives
- Weight j
  - Derivatives small (rare feature)
    - Larger learning rate to give them more importance
  - Derivatives large, small learning rate
- No longer moving in the gradient direction
- Learning rate always goes to zero
  - Not necessarily desirable

$$\lambda_t^j = \frac{\alpha}{\sqrt{\sum_{i=1}^{t-1} \nabla l(w_i)_j^2}}$$

Auxiliary Formula

$$u_t = \vartheta u_{t-1} + (1-\vartheta) z_t$$

$$<=>$$

$$u_t = (1-\vartheta)(z_t + \vartheta z_{t-1} + \vartheta^2 z_{t-2} + \cdots + \vartheta^t z_0)$$

- Exponential decay
- Can be easily tracked recursively

Adaptive: RMSprop

- Similar but weigh more recent iterates more
- Two parameters
  - Alpha as numerator
  - Theta for decay
- Updates can be made recursively

$$z_t^j = \nabla l(w_t)_j^2$$

$$lr_t^j = \frac{\alpha}{\sqrt{u_t^j}}$$