ECE7121 Learning-based control – 2025 Fall

Imitation learning

INHA UNIVERSITY

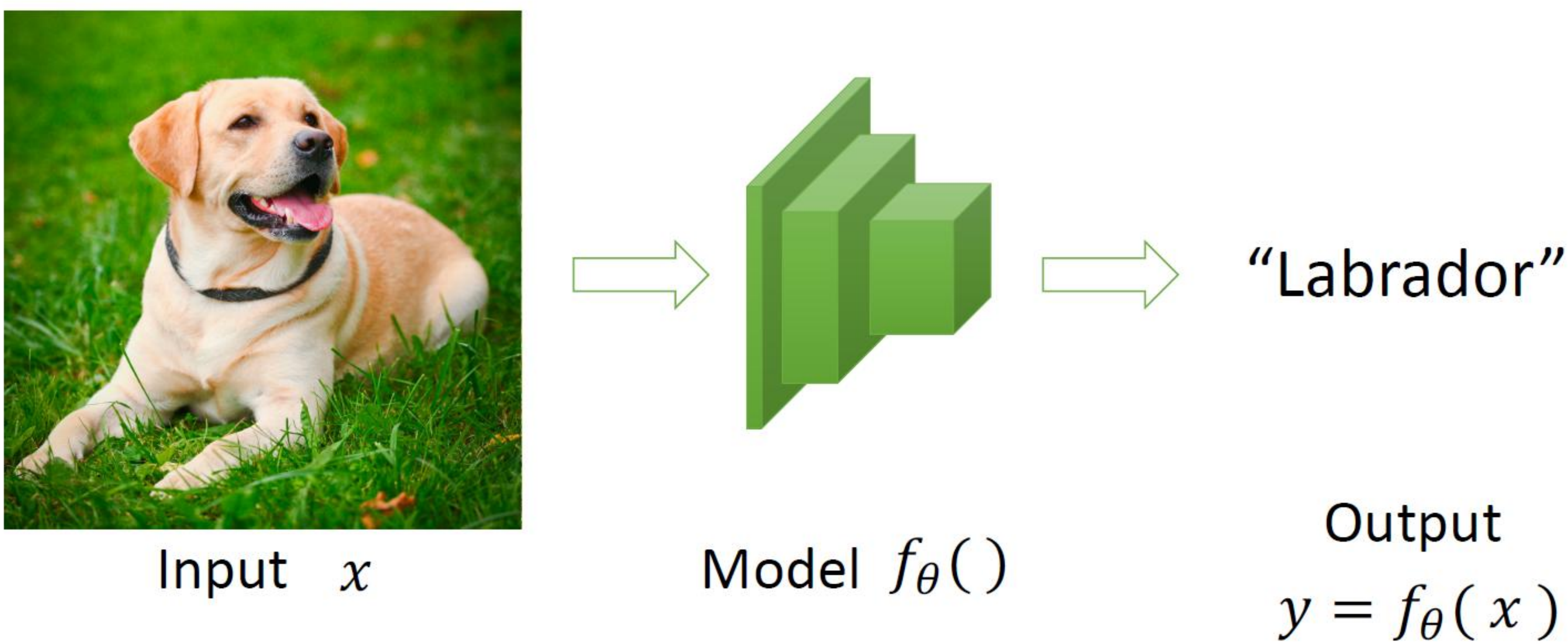
Overview

> Basics of deep learning
  - neural networks
  - supervised learning
  - optimization

> Imitation learning
  - issues
  - Dagger
  - multi-modality

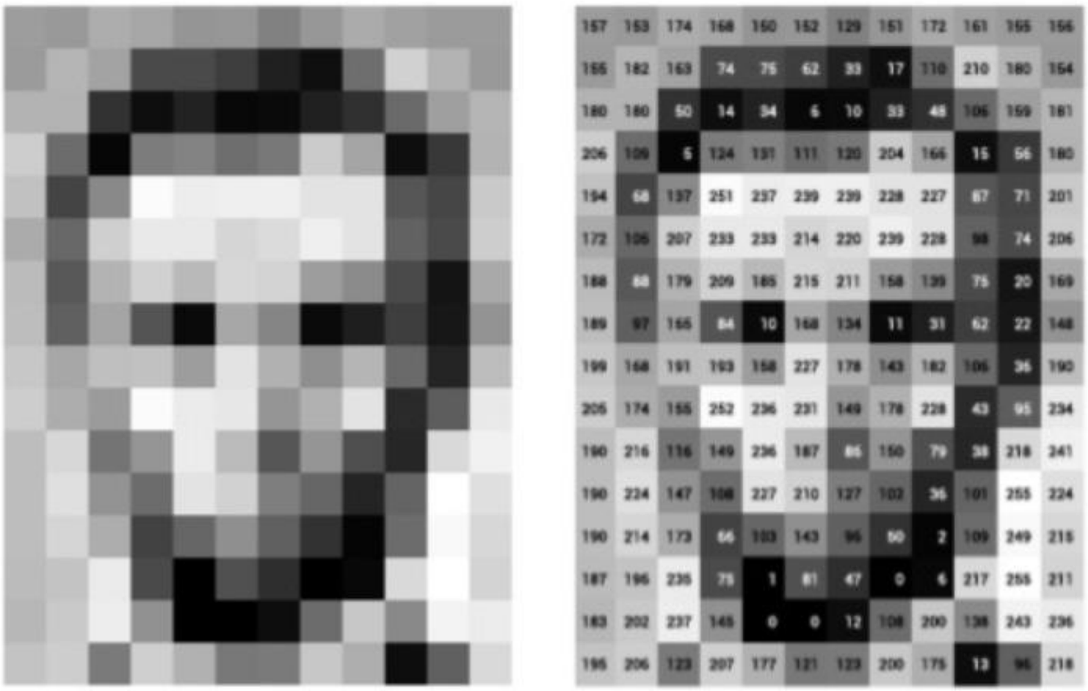
Supervised learning

- Learn a function $f: X \rightarrow Y$ from an input space X to an output space Y
- Ideally, such learned function f will perform well on the test data



Supervised learning

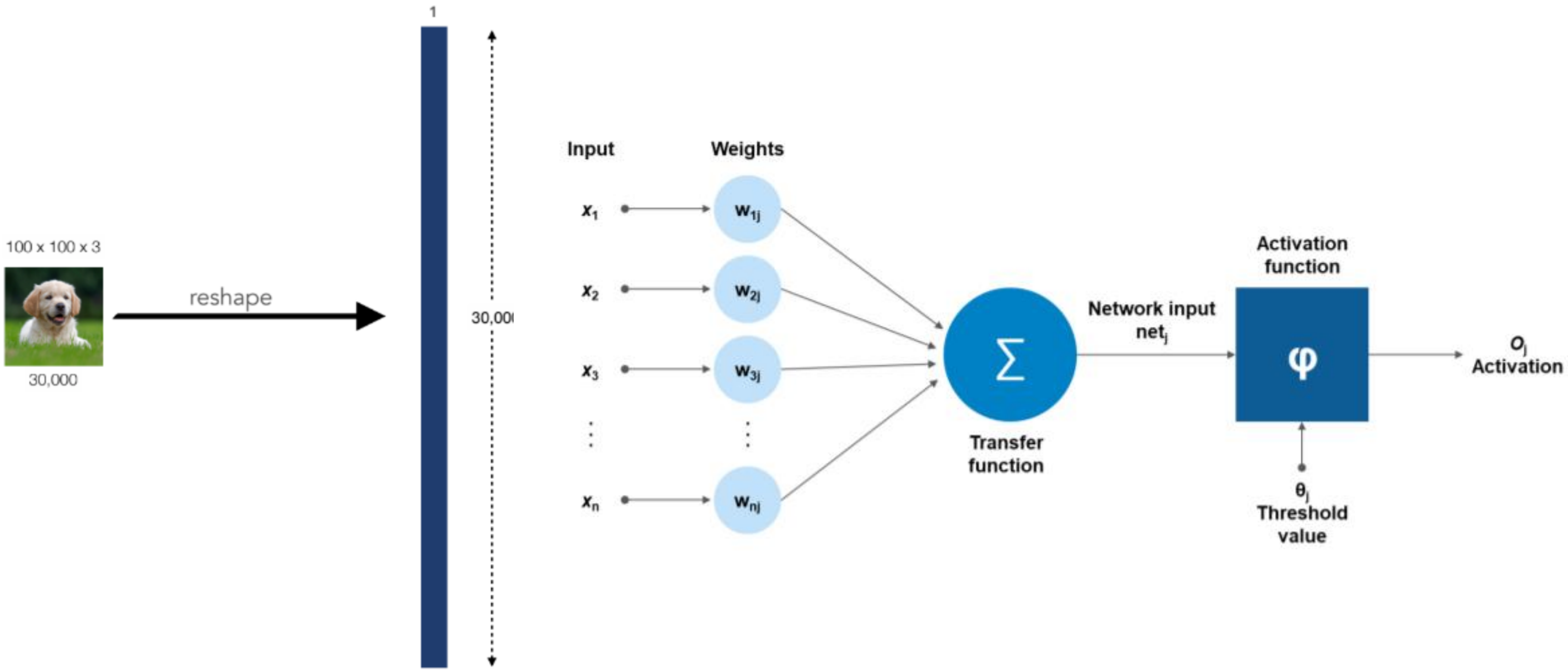
> Data representation of the image



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

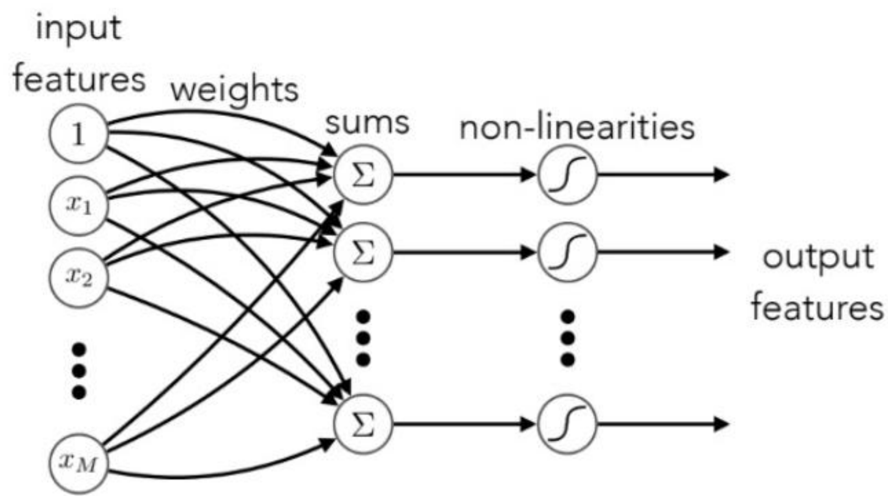
Artificial neuron model

> One neuron

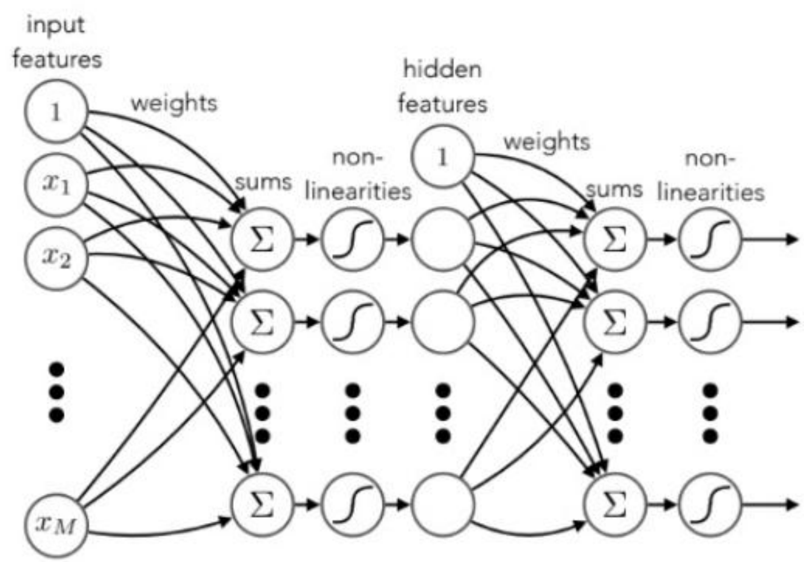


Artificial neuron model

- One layer



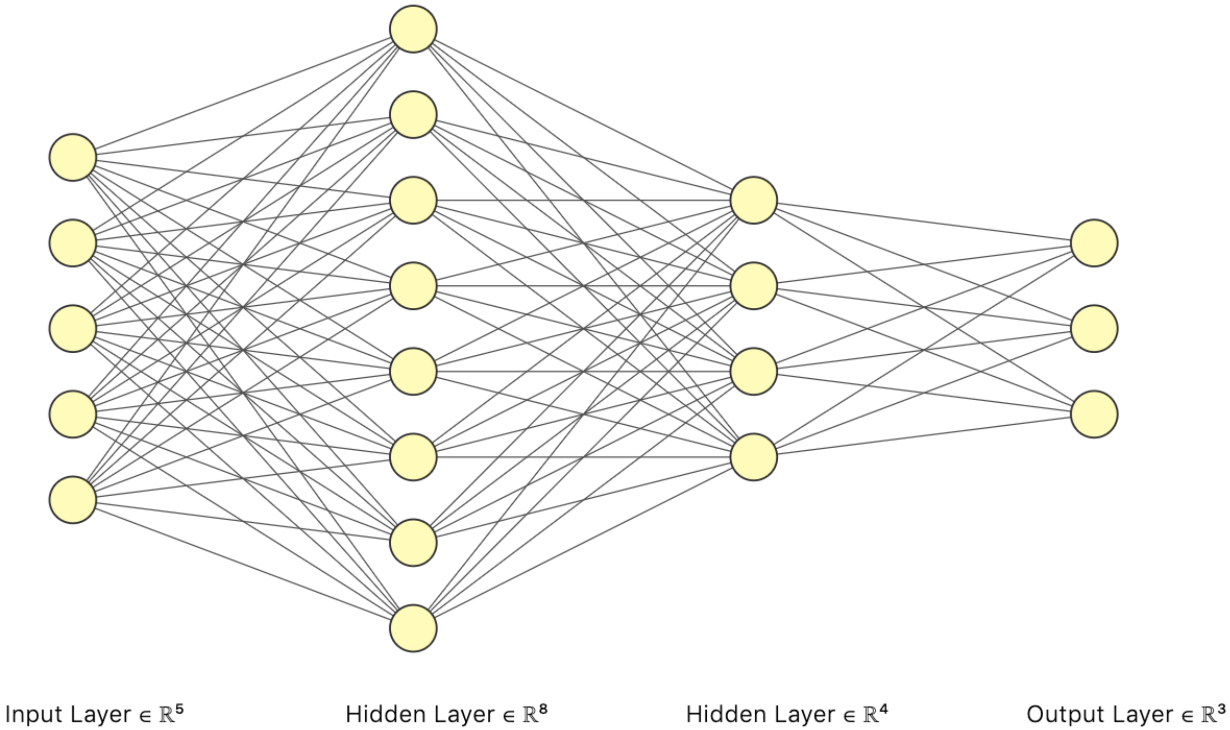
- Multi-layer



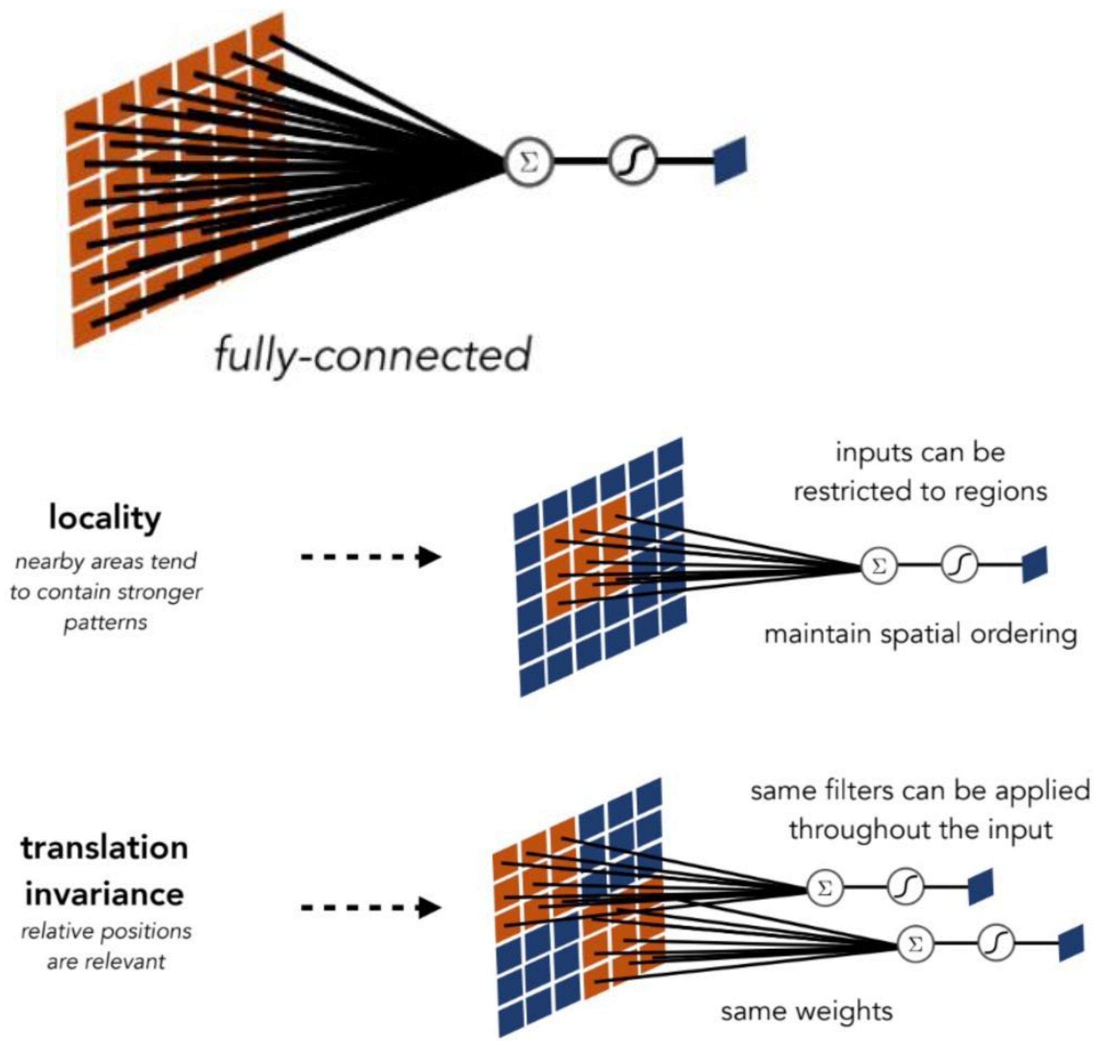
Artificial neuron model

> Number of parameters
  - for first operations: 5*8 + 8 = 48
  - for second operations: 8*4+4 = 36
  - for third operations: 4*3+3 = 15

Total: 99 parameters



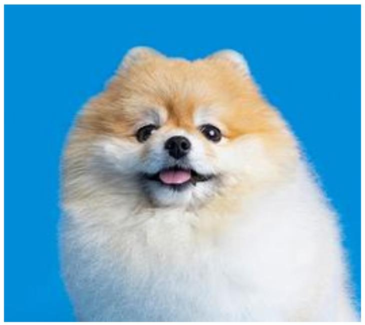
Convolutional neural networks



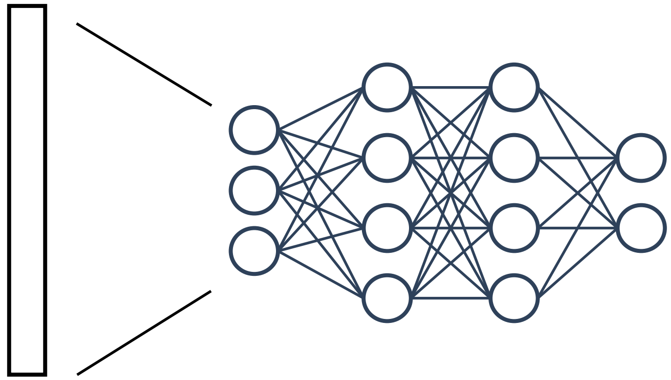
Convolutional neural networks

> MLP vs CNN

32



32

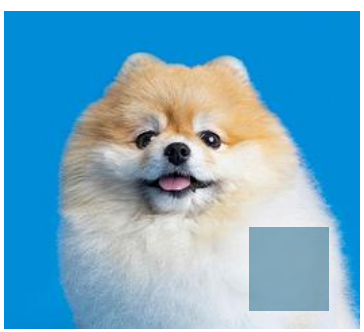


Flatten
32*32*3 =
3072

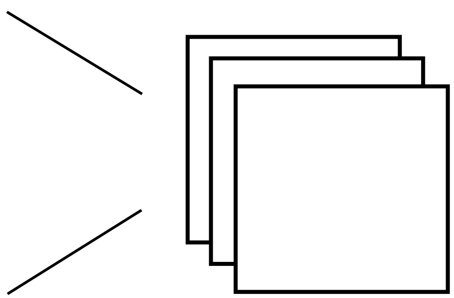
Layer 1 (256) =
3072*256

Layer 2 (64) =
256*64

32



32



5 by 5 kernel

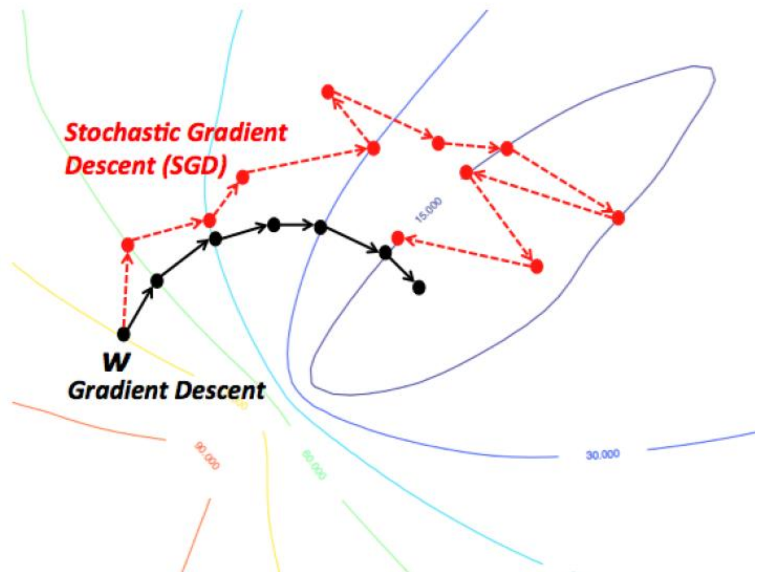
8 channels = 5*5*3*8=600

Optimization

> Model: try to match the result of NN layers with the label
  - goal: $y \approx f(x_i|w)$
  - loss: $\mathrm{L}(\mathrm{y}, \mathrm{y}') = (\mathrm{y} - \mathrm{y}')^2$ (squared loss)
  - objective: $= \mathrm{argmin}_w L_N(w) = \sum_{i \in D_{train}} L(y_i, f(x_i|w))$

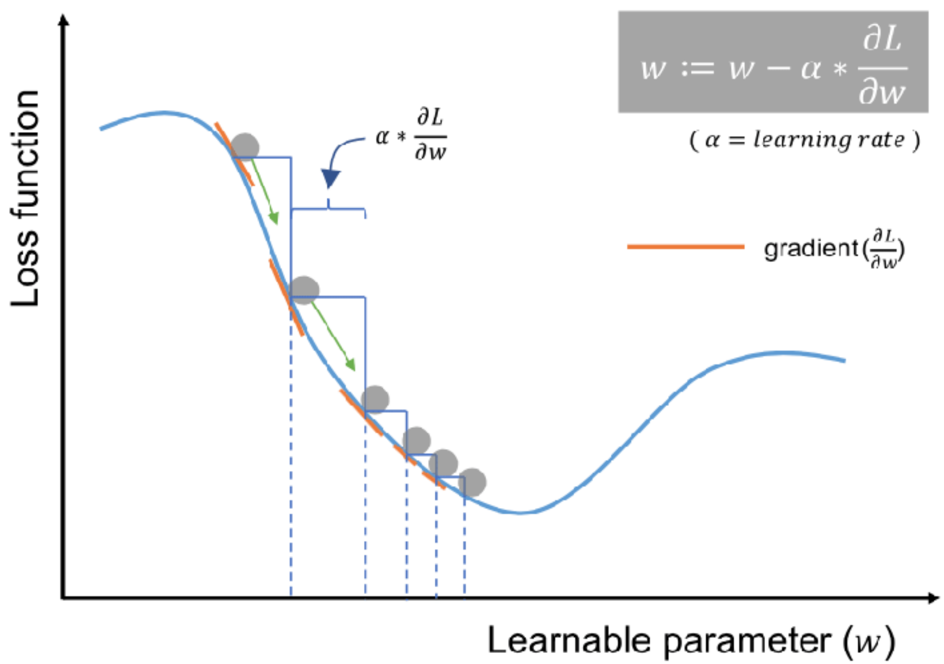
> Gradient descent (GD)
  - $w \leftarrow w - \eta \nabla_{\mathrm{w}} \sum_{i \in D_{train}} L(y_i, f(x_i|w))$
  - expensive if the training dataset is large

> Stochastic gradient descent (SGD)
  - $w \leftarrow w - \eta \nabla_{\mathrm{w}} \sum_{i \in Batch} L(y_i, f(x_i|w))$
  - only needs access to one batch at a tim
  - can be parallelized



Gradient descent

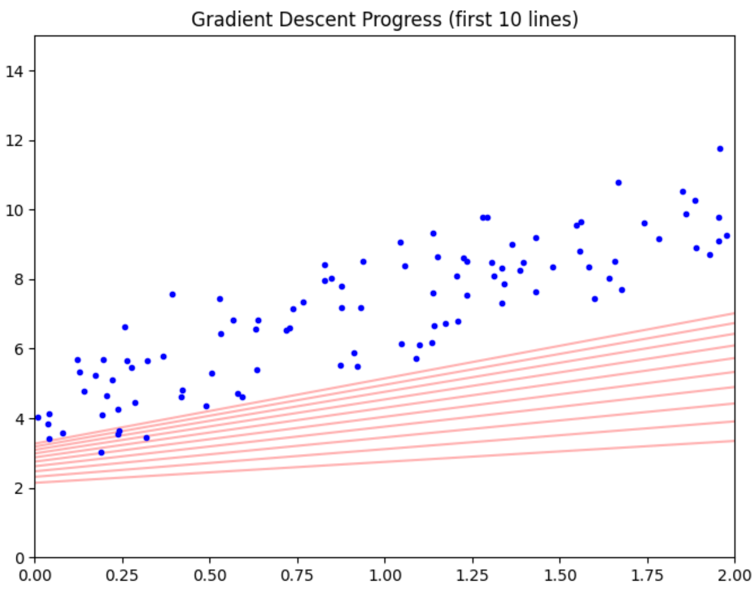
- $x_{t+1} = x_t - \alpha_t \nabla f(x_t)$
- Gradient descent is steepest descent method
  - direction $\nabla f(x)$ gives greatest reduction in $f(x_t)$ per unit change in x
  - learning rate determines how far we move in that direction

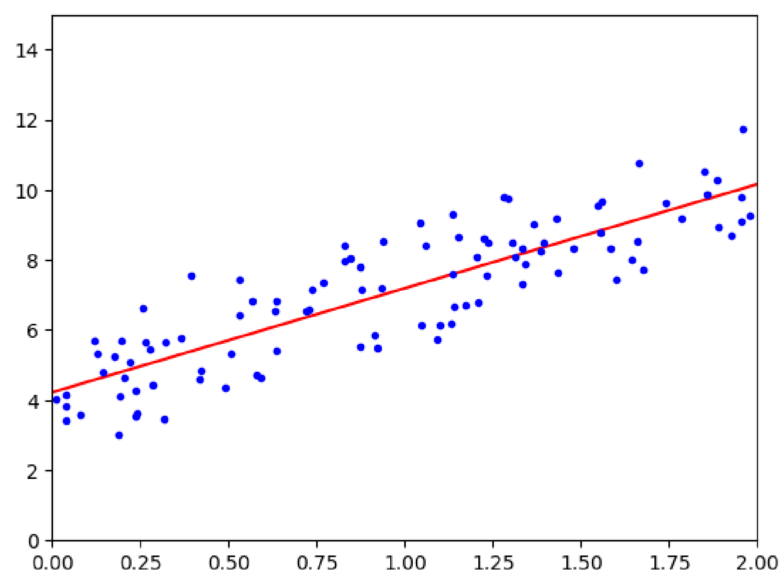


Gradient descent

> Linear regression $h_\theta(x) = \theta_0 + \theta_1 x_1$

- $J(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$
- $\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial\theta_j} J(\theta)$, α is the learning rate





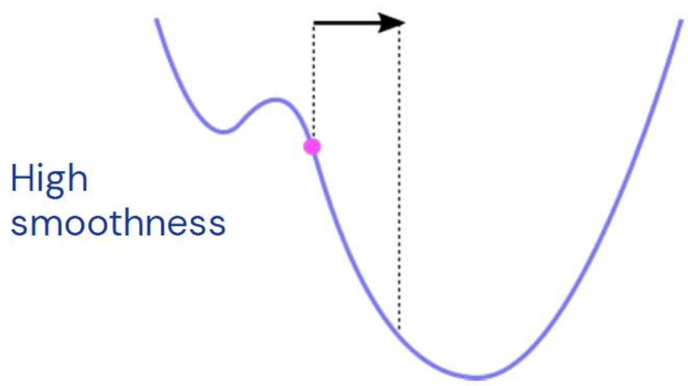
Gradient descent

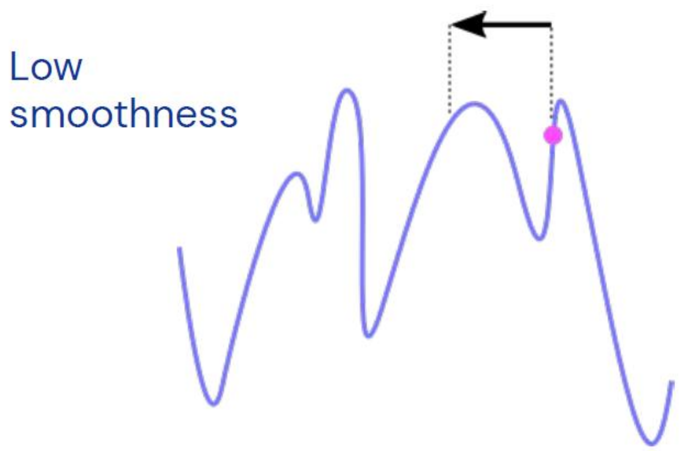
> 1^{st} order Tayler series for $f(x)$ around current x is

$$f(x+d) \approx f(x) + \nabla f(x)^\top d$$

- for small enough d, this will be a reasonable approximation

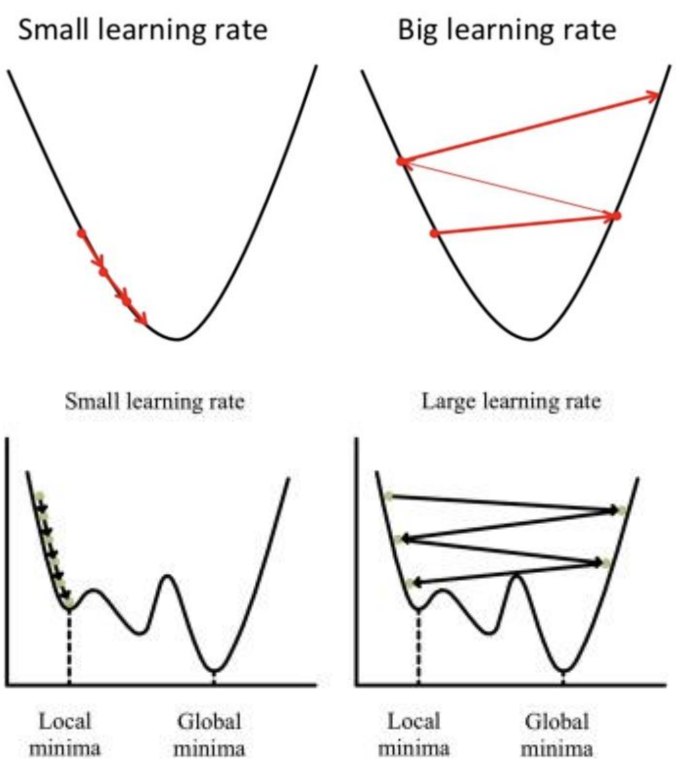
> If $f(x)$ is sufficiently smooth, and learning rate is small, gradient will keep pointing down-hill over the region

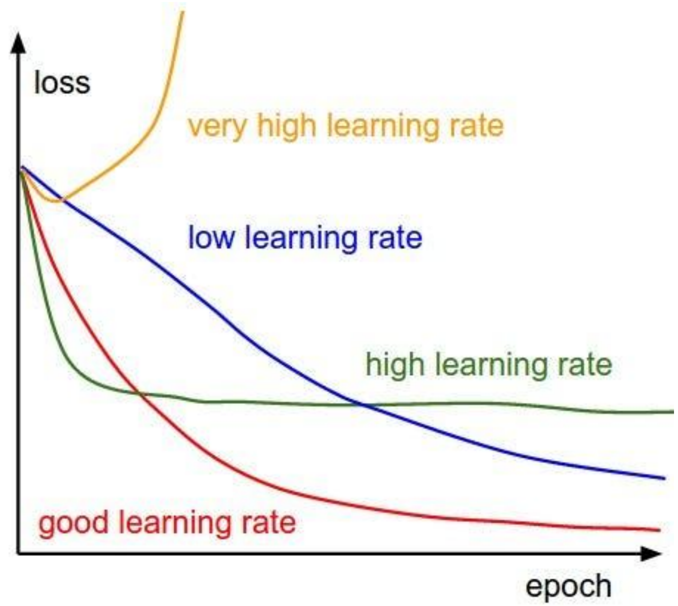




Challenges

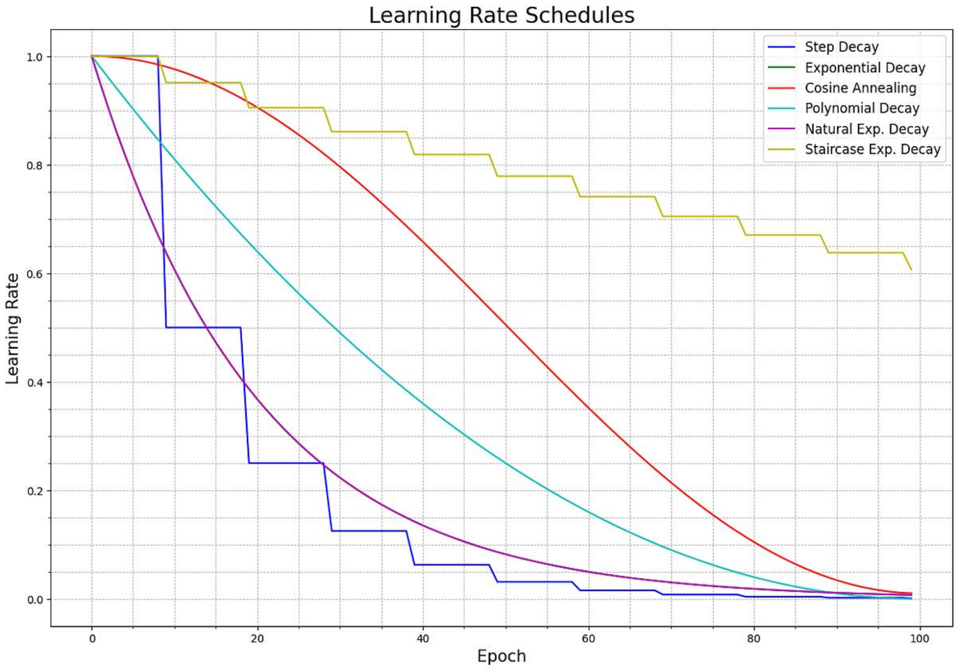
- Choosing a proper learning rate can be difficult
  - a small learning rate lead slow convergence, while a large learning rate hinder convergence
  - a small learning rate may get stuck in local minima





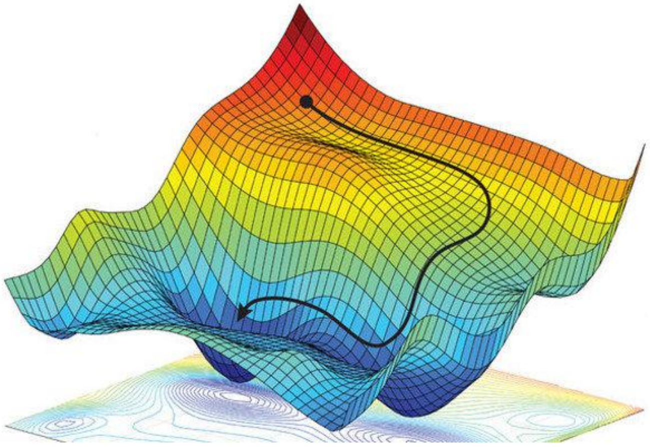
Challenges

- Choosing a proper learning rate can be difficult
  - simple solution is scheduling the learning rate
  - $\alpha_t = \alpha_0 \beta^t, \beta \in [0,1]$
  - reducing the learning rate according to a predefined schedule (have to be defined in advance, thus unable to adapt to a dataset)

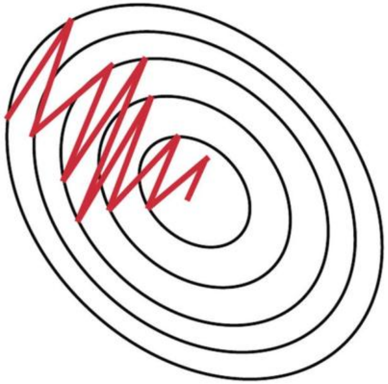


Optimization

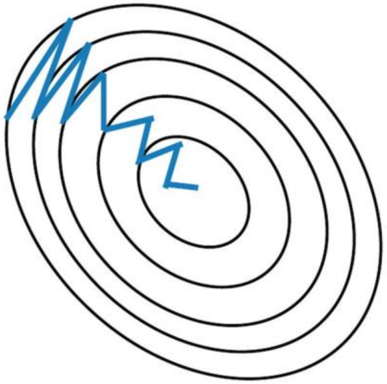
> Beyond SGD
  - non-convex problems are hard to optimize
  - multiple local minima



> Many optimization methods
  - Momentum, Adagrad, RMSProp, Adam
  - AdamW(2019): transformer, Lion(2023):light



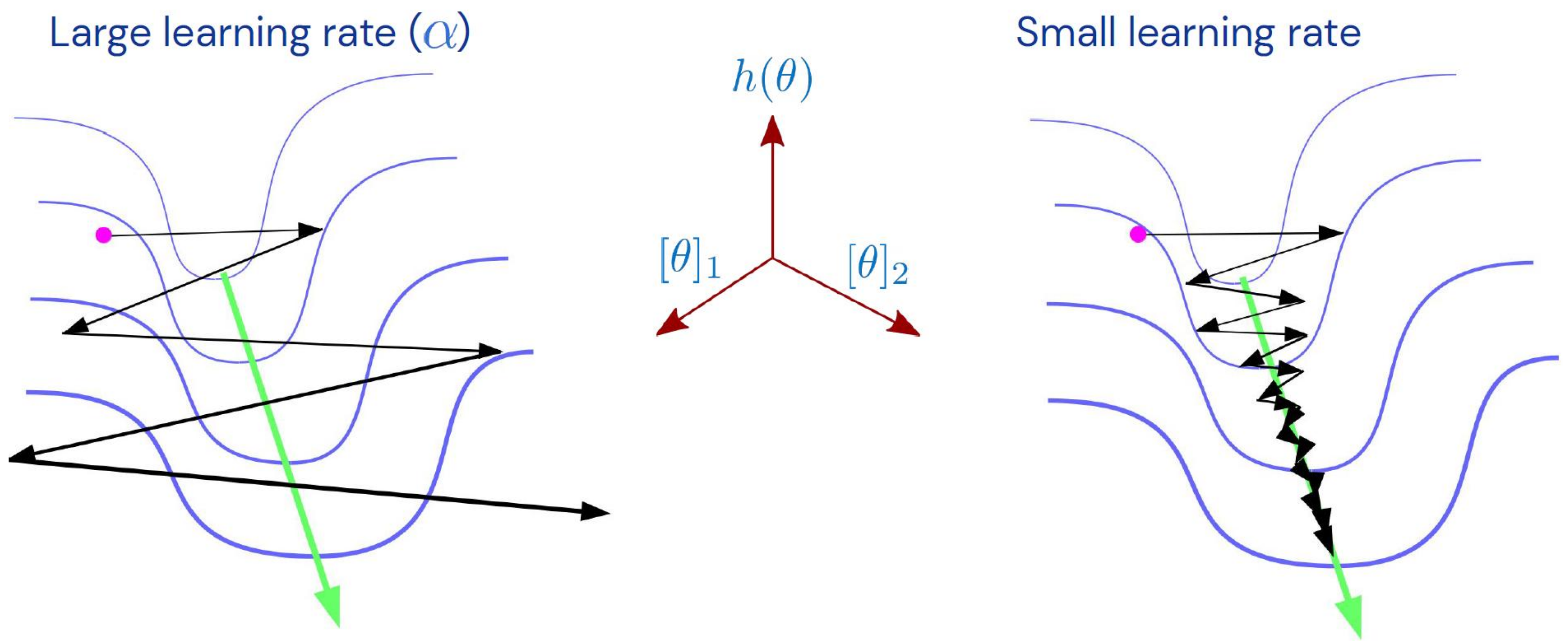
Stochastic Gradient Descent **withhout** Momentum



Stochastic Gradient Descent **with** Momentum

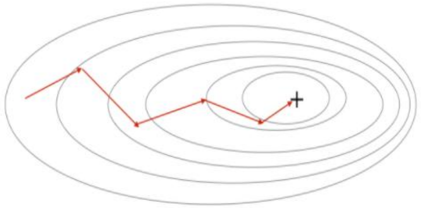
Challenges

- Gradient direction is not optimal
  - Steepest gradient is not the most efficient way to the minimum
  - e.g., 2D narrow valley example



Momentum method

- Motivation: gradient has a tendency to flip back and forth as we take steps when the learning rate is large



- Key idea: accelerate movement along directions that point consistently down-hill across many consecutive iterations

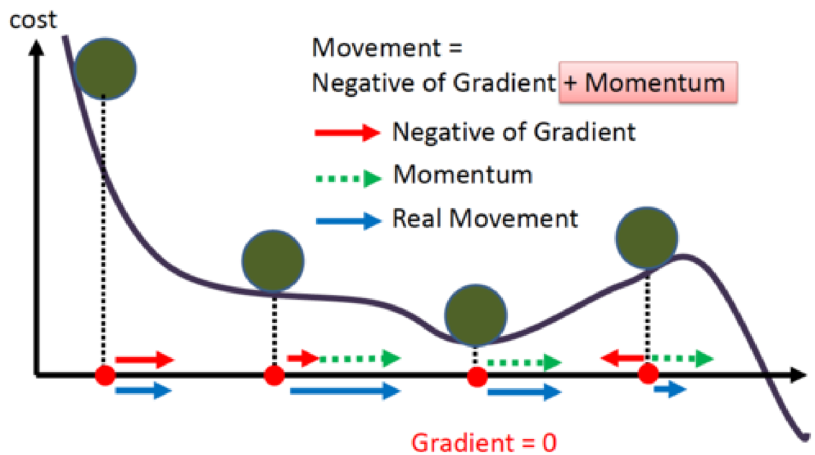
$$x_{t+1} = x_t + \alpha_t v_{t+1}$$

$$v_{t+1} = \gamma v_t - \nabla f(x_t), \quad v_0 = 0$$

γ is momentum constant (usually 0.9)

Momentum method

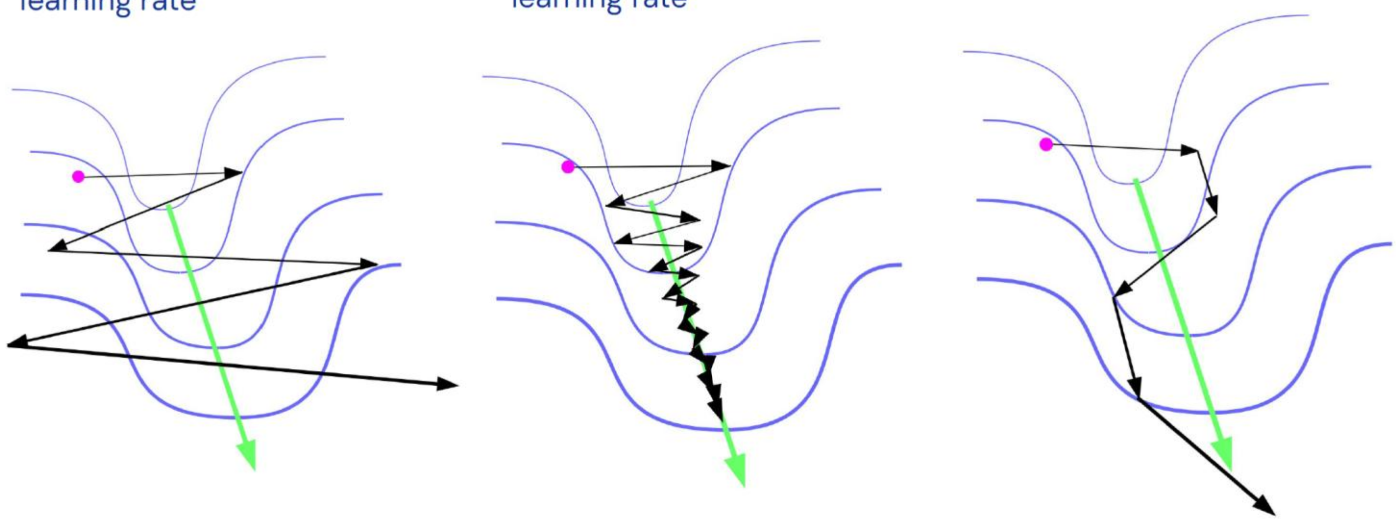
>



Gradient descent with large learning rate

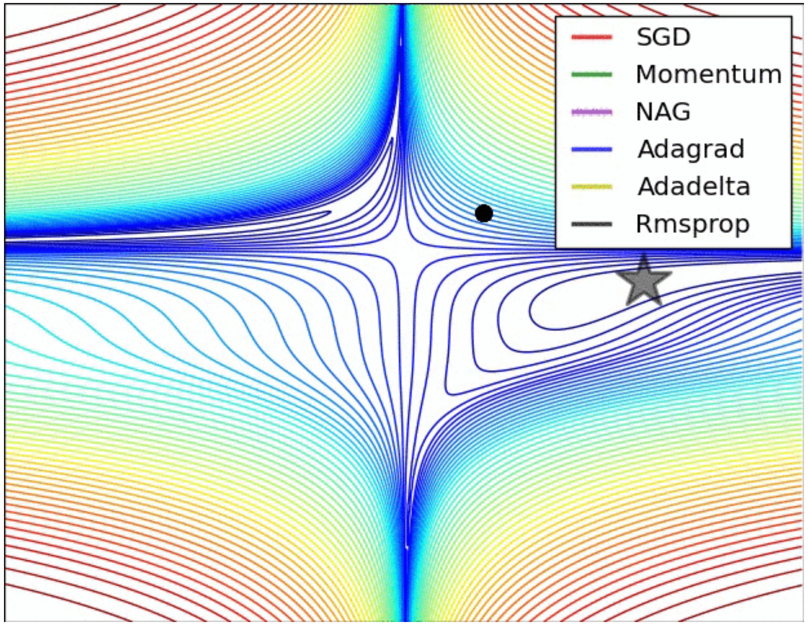
Gradient descent with small learning rate

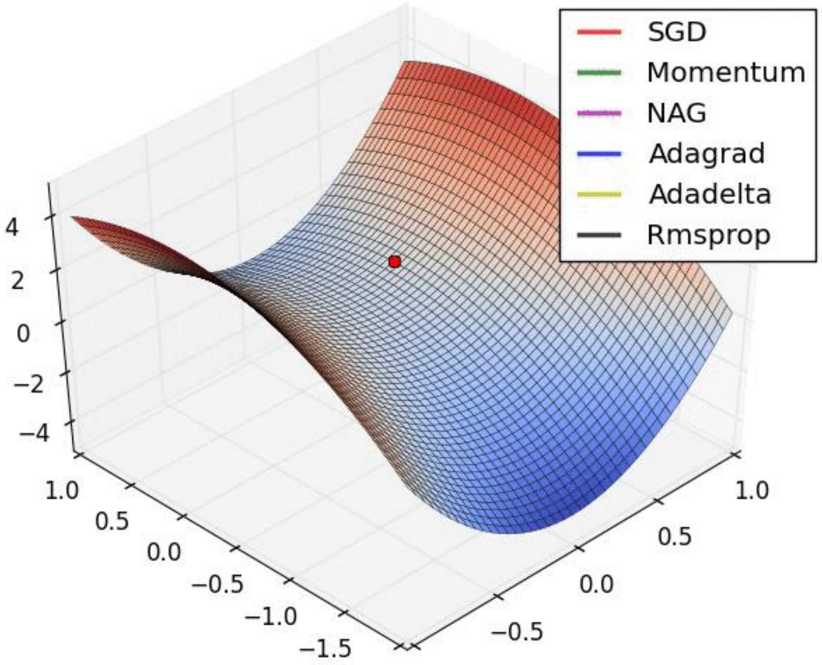
Momentum method



Optimizers

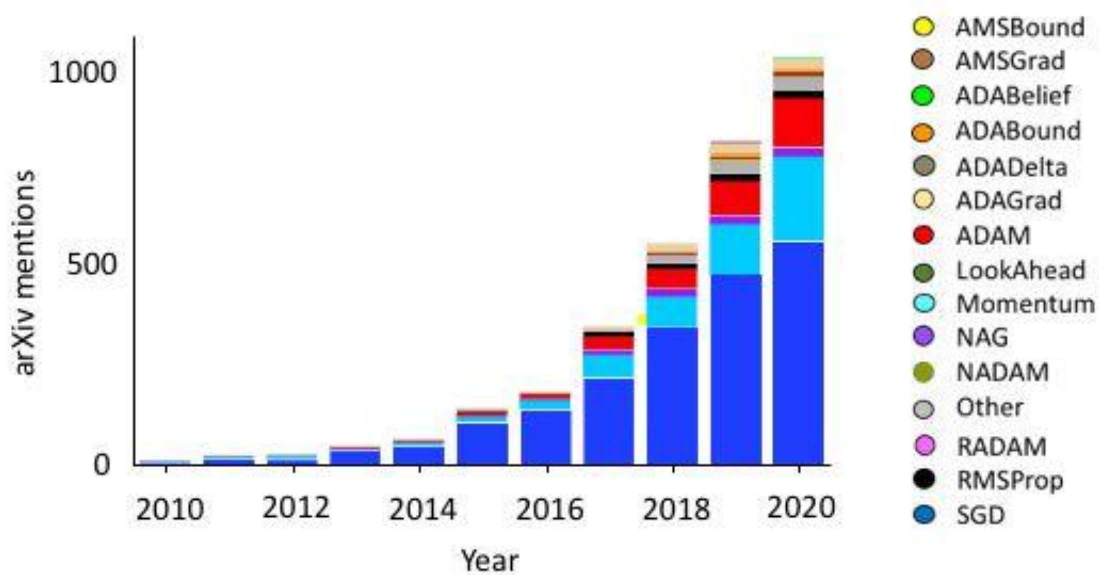
> Convergence performance





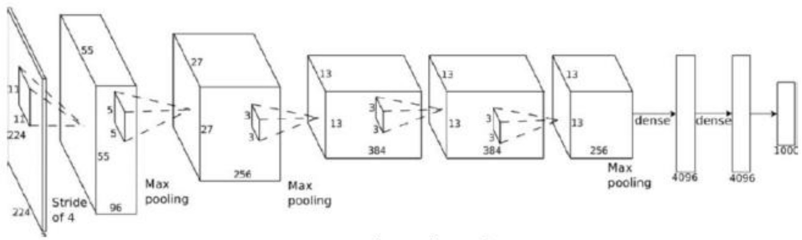
Optimizers

> Which optimizer to use?
  - Adam / AdamW are dominant and default choices
  - In cases where another optimizer did better than Adam, it was usually RMSProp or NAG
  - Adafactor, Lion, Signum are promising for LLMs
  - SGD (+momentum) is good for vision tasks, but requires careful tuning



Imitation learning

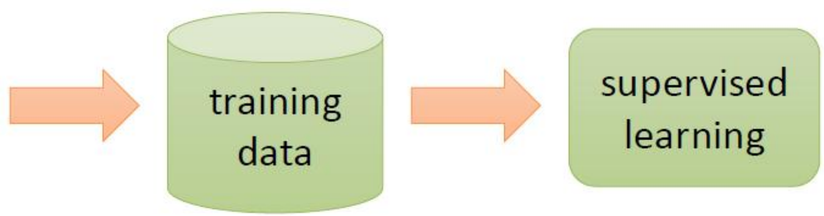
> Given some expert trajectories, the agent is trained to copy the expert



$\mathbf{o}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\mathbf{a}_t$

$\mathbf{o}_t$

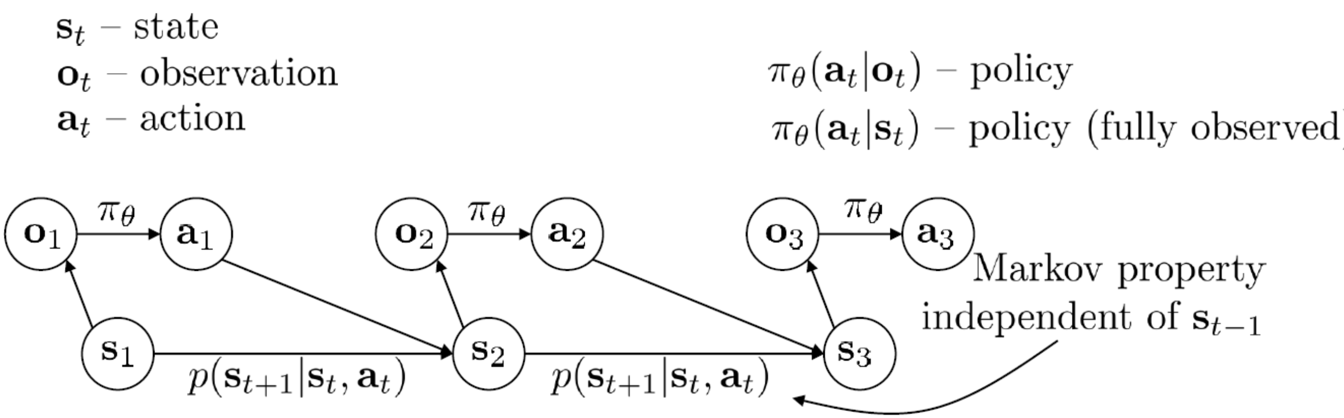
$\mathbf{a}_t$



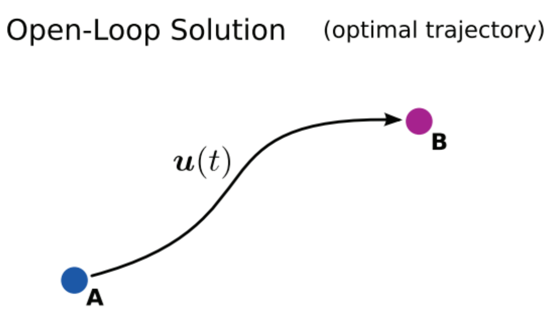
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

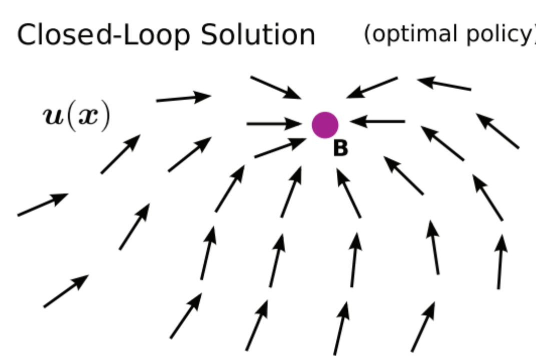
Terminology

> Markov property: previous state can tell all information
  - MDP / POMDP



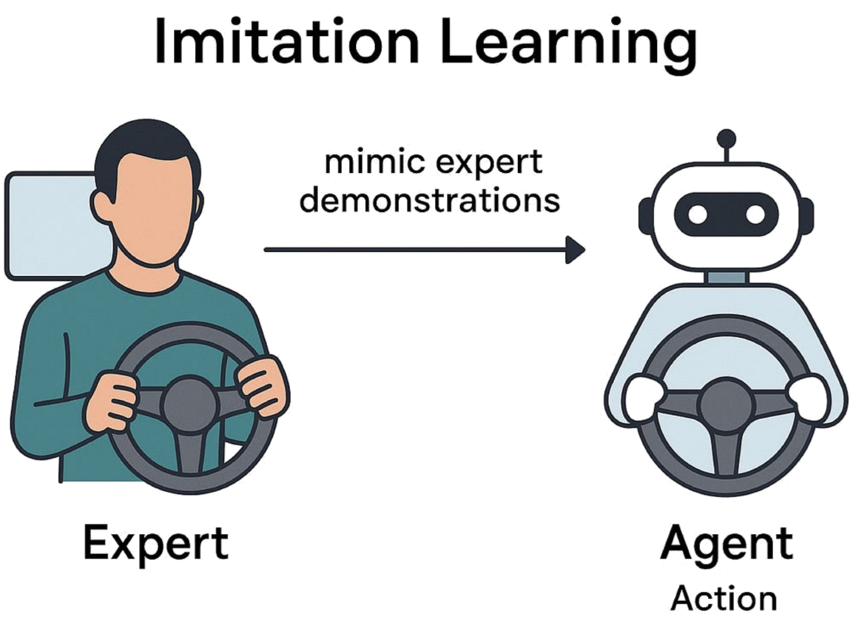
trajectory $\tau = (s_1, a_1, s_2, a_2, \dots, s_T, a_T)$, sequence of states/observations and actions





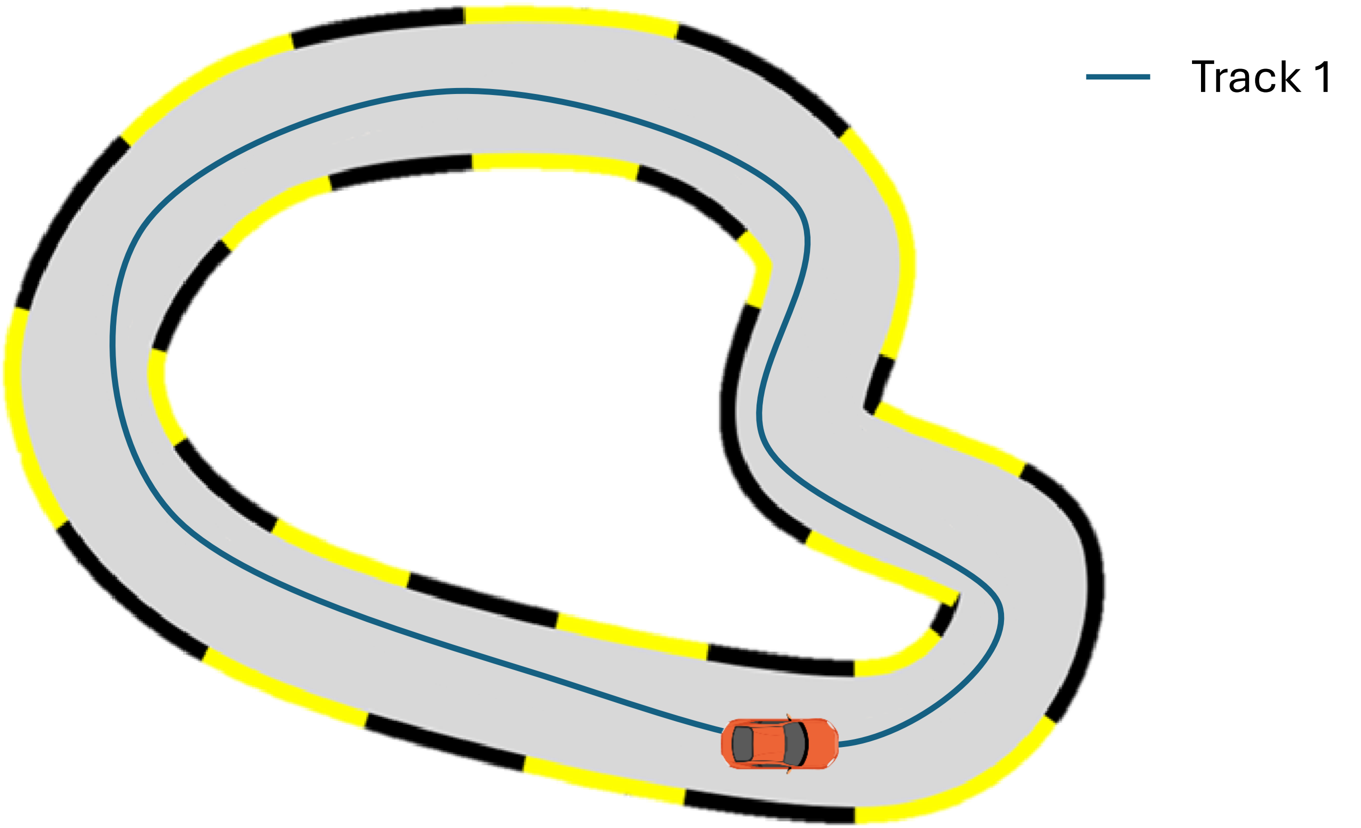
IL - autonomous driving

- Dataset from human drivers
- Sensor readings + steering commands



IL - autonomous driving

- Collect an expert dataset
- Imitate $\hat{a} = \pi_\theta(s)$, minimize $|a - \hat{a}|^2$



IL - autonomous driving

- We don’t know how to behave for the unexperienced state
- Can we cover all possible states?

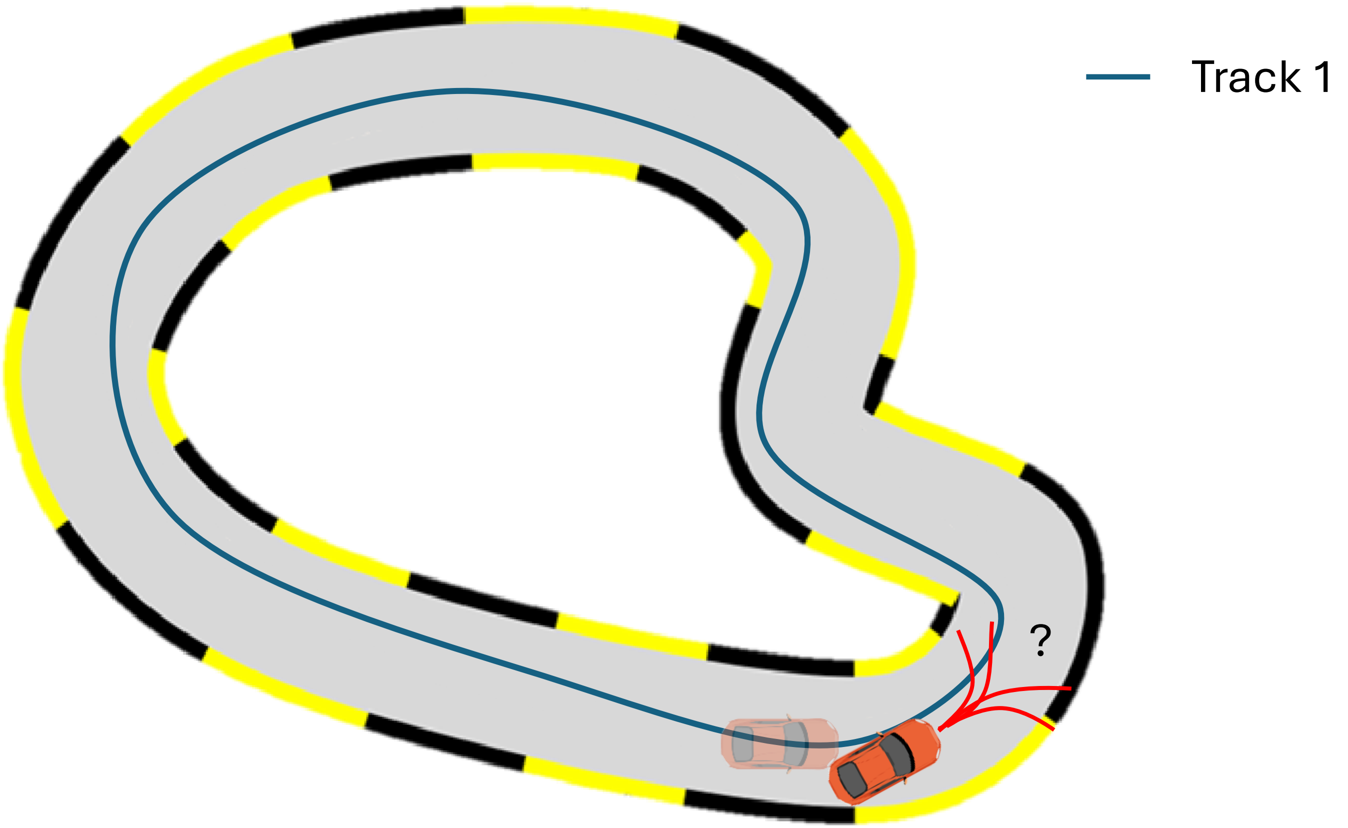
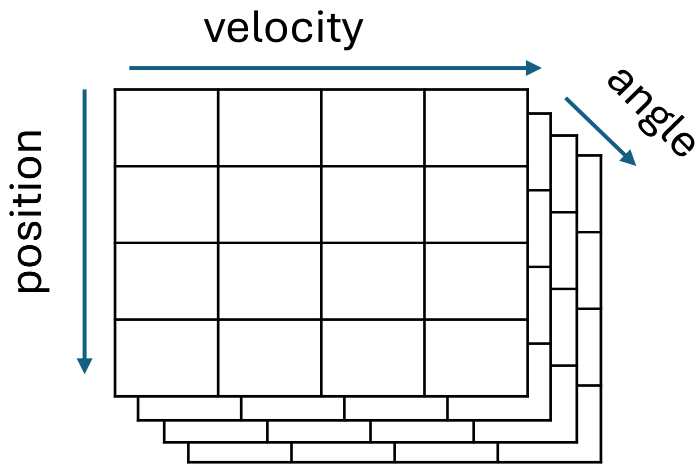
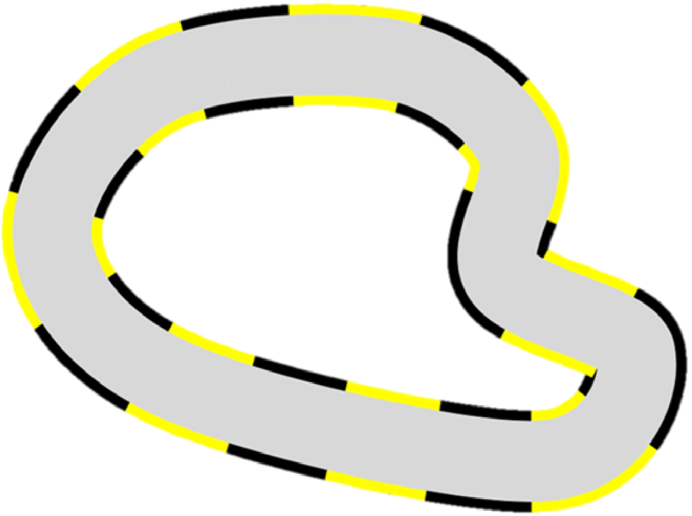


Table-look up method

> $\pi: s \in \mathbb{R}^n \rightarrow a \in \mathbb{R}^m$
  - All possible states (N-d array) → expert action

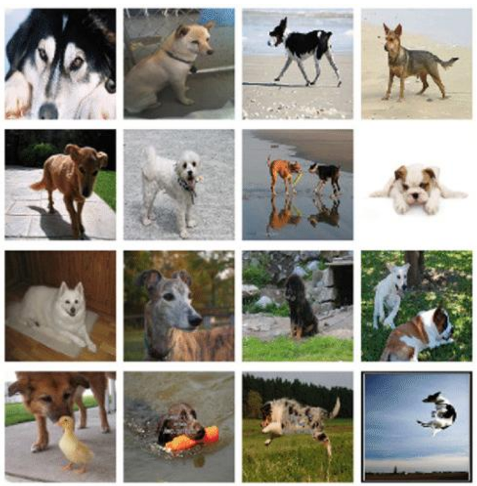




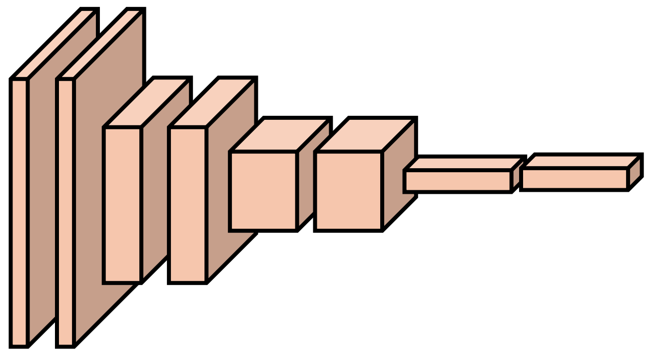
  - State has continuous value → binning

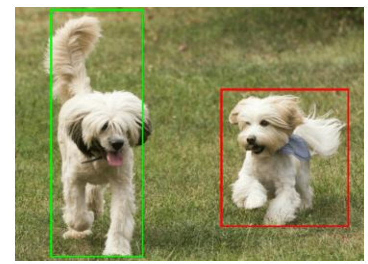
Why not table-look up?

- The role of learning is generalization! (interpolation)
  - Neural network is a continuous function



Tons of dog images



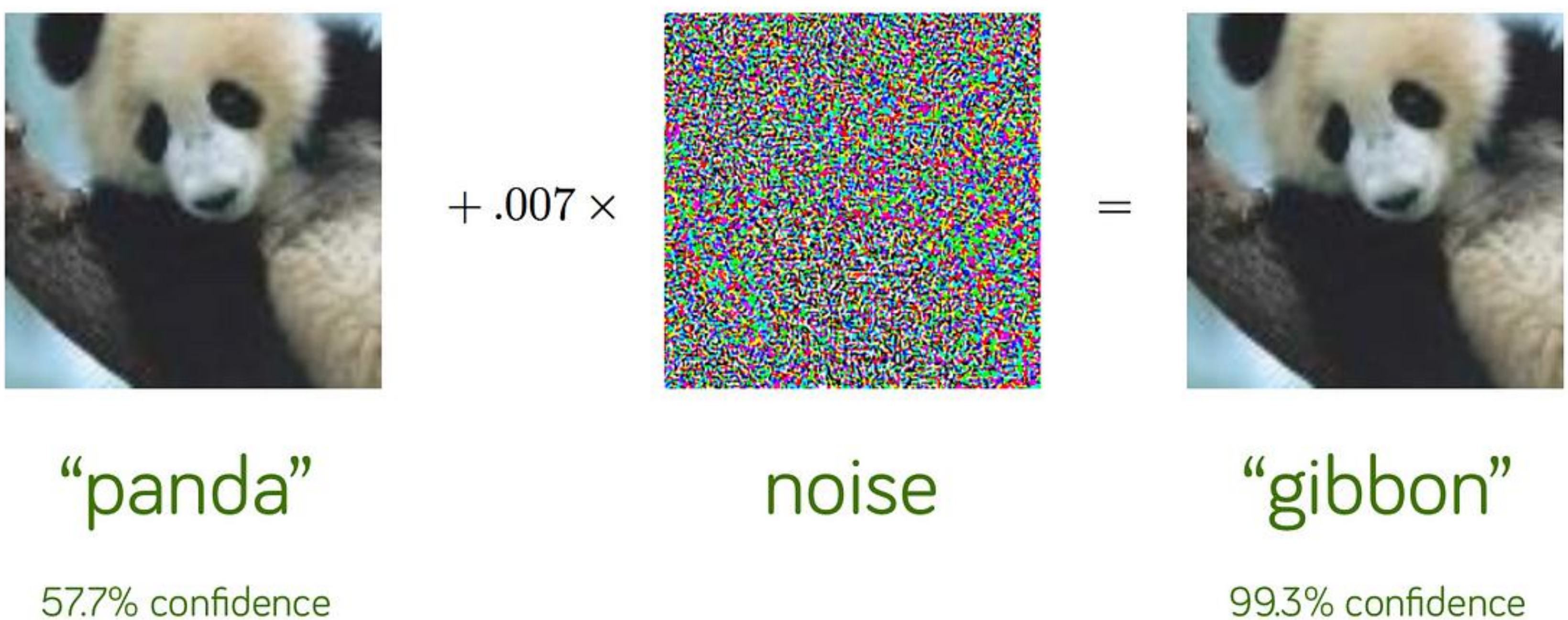


Weakness of neural networks

- Number of possible positions is about 10^{170}
- Number of sand particles on Earth is about 10^{20}
- KataGo defeated by amateur players through adversarial attacks

Weakness of neural networks

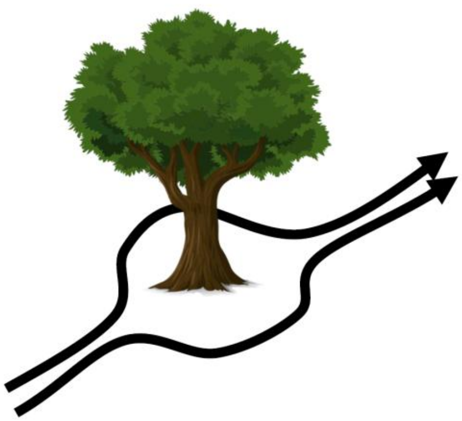
> Adversarial attack
  - Neural network is fragile!

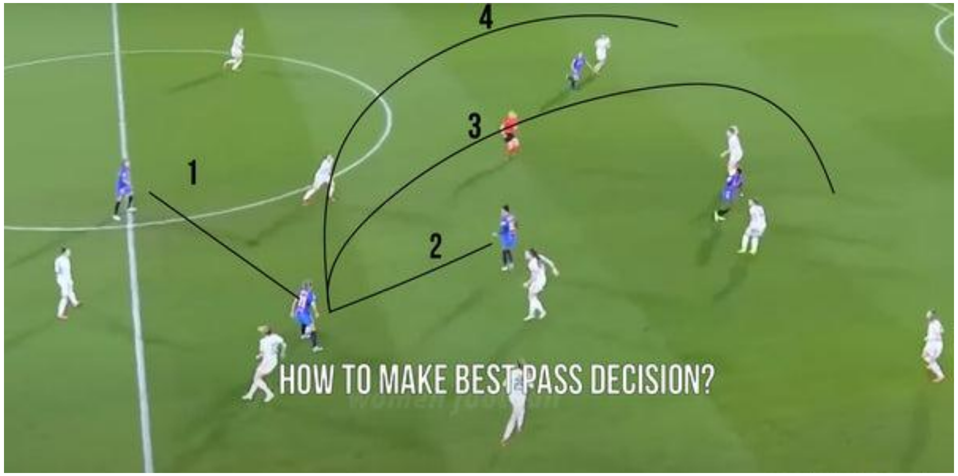


Issues in imitation learning

> We can't cover the all-possible state space
  - However, we will always encounter the new situations
  - What can we do?
  - Collect more data!

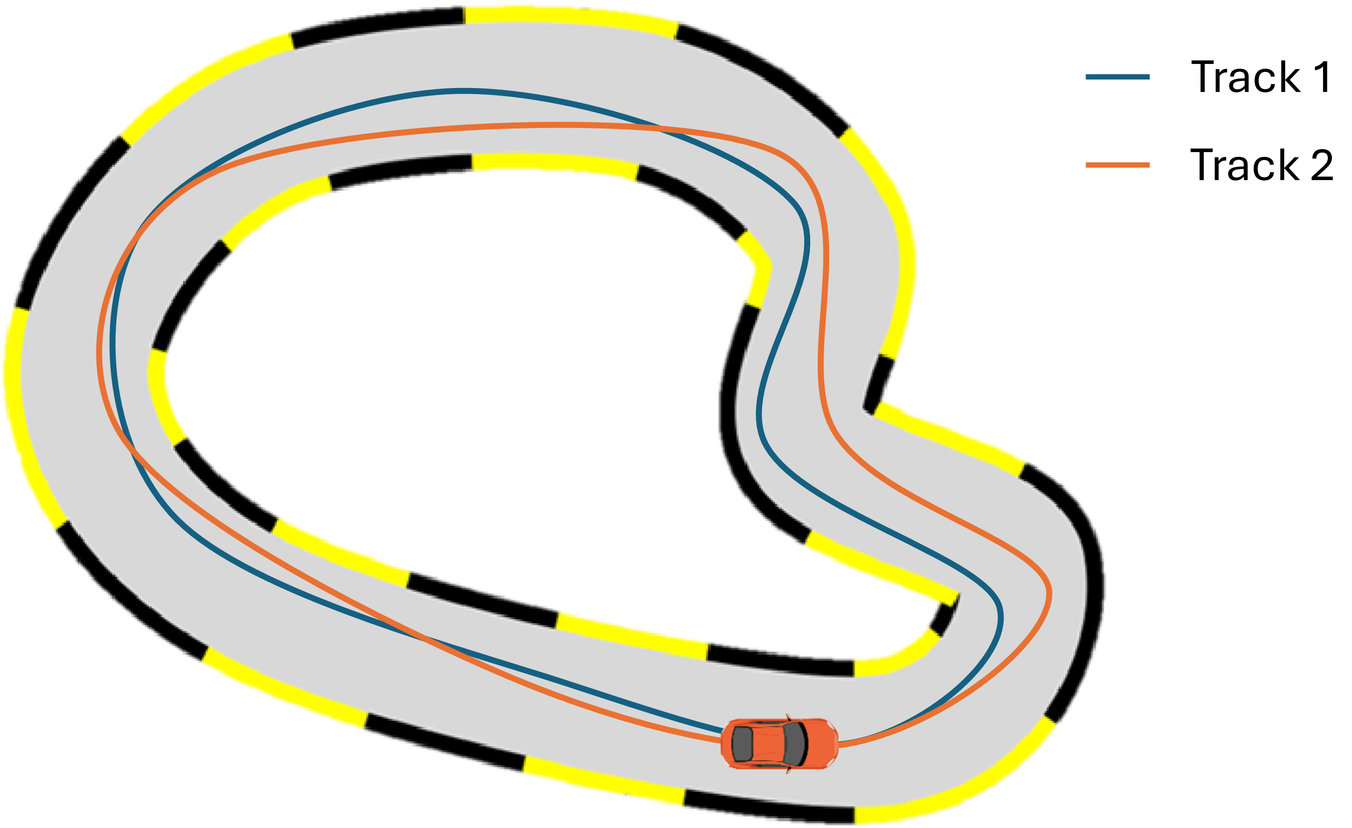
> Expert policy might not be optimal or there would be multiple optimal actions





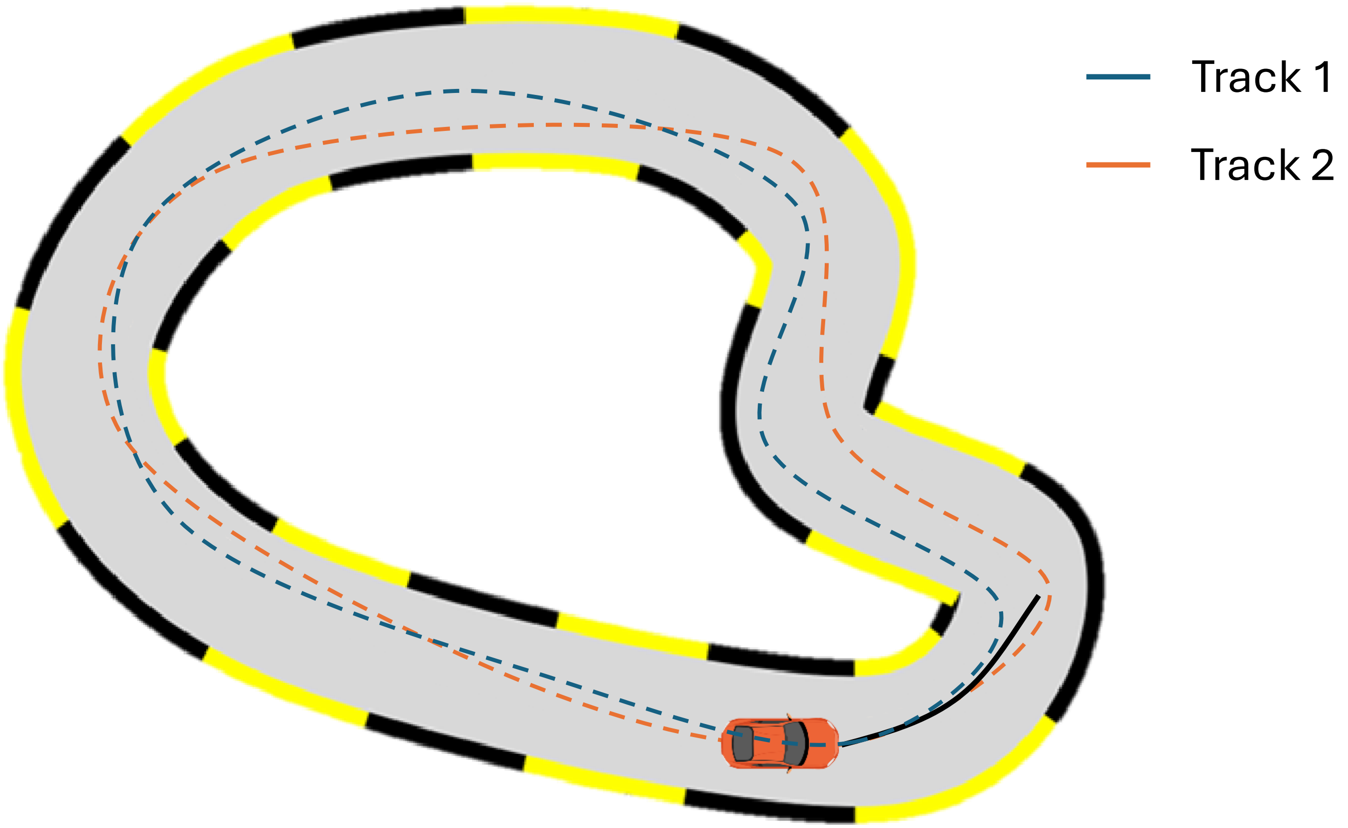
IL - autonomous driving

- Collect a lot of driving datasets
- Imitate $\hat{a} = \pi_\theta(s)$, minimize $|a - \hat{a}|^2$



IL - autonomous driving

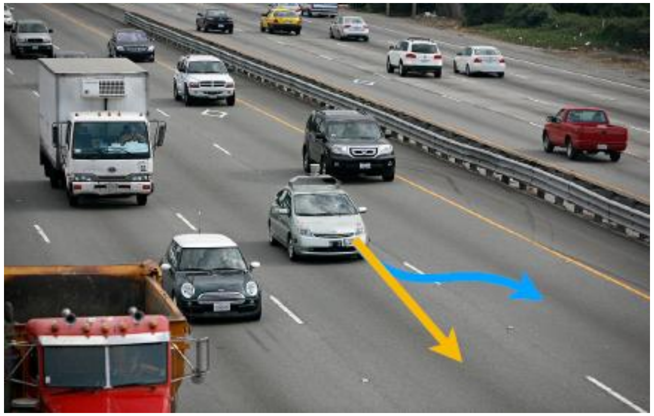
- Collect a lot of driving datasets
- Imitate $\hat{a} = \pi_\theta(s)$, minimize $|a - \hat{a}|^2$

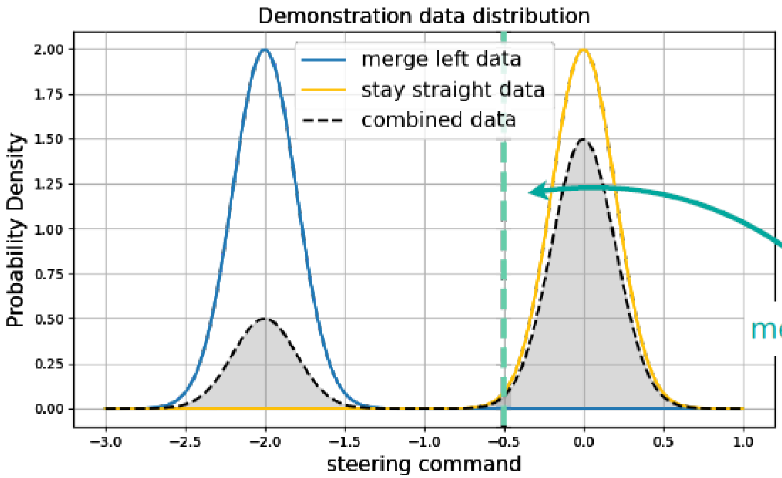


IL - autonomous driving

- Collect a lot of driving datasets
- Imitate $\hat{a} = \pi_\theta(s)$, minimize $|a - \hat{a}|^2$
- When data collected by multiple people

Question: what might policy trained with ℓ_2-regression do?

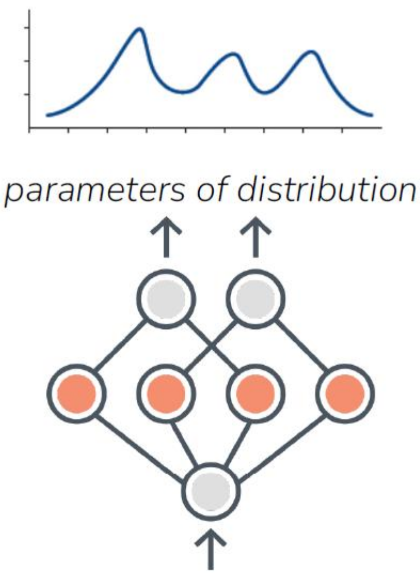




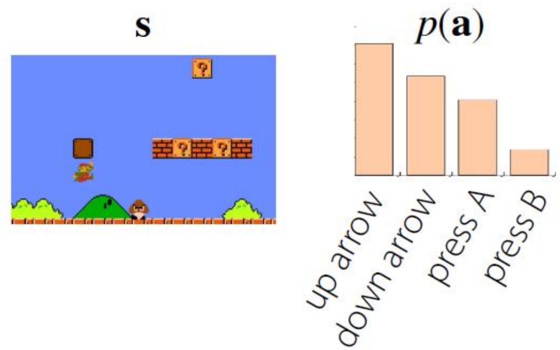
mean of data

Learning distributions

- Learning distributions with neural networks
  - NN expressivity is often distinct from distribution expressivity



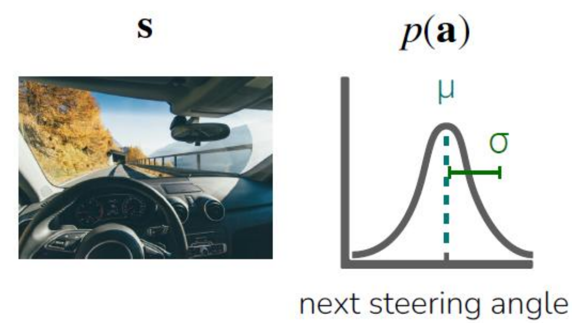
1D discrete actions



Neural net outputs $p(\text{up}), p(\text{down}), \ldots$ represent categorical distribution.

Maximally expressive

Continuous actions

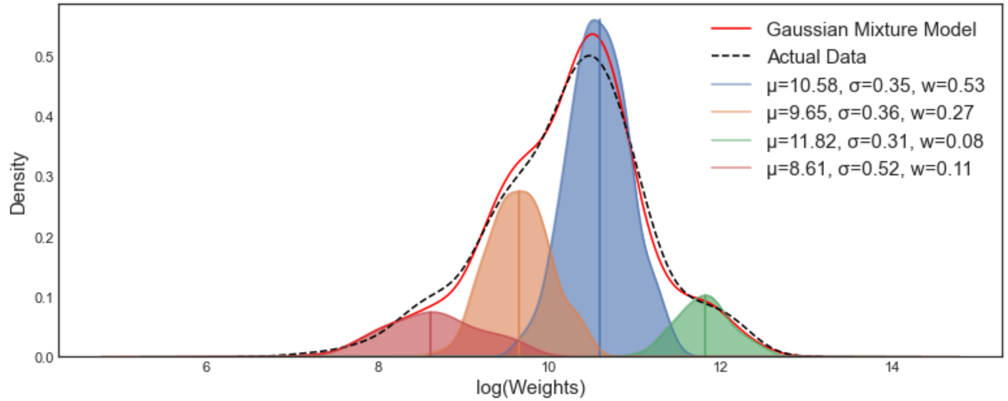


Neural net outputs μ, σ to represent Gaussian distribution.

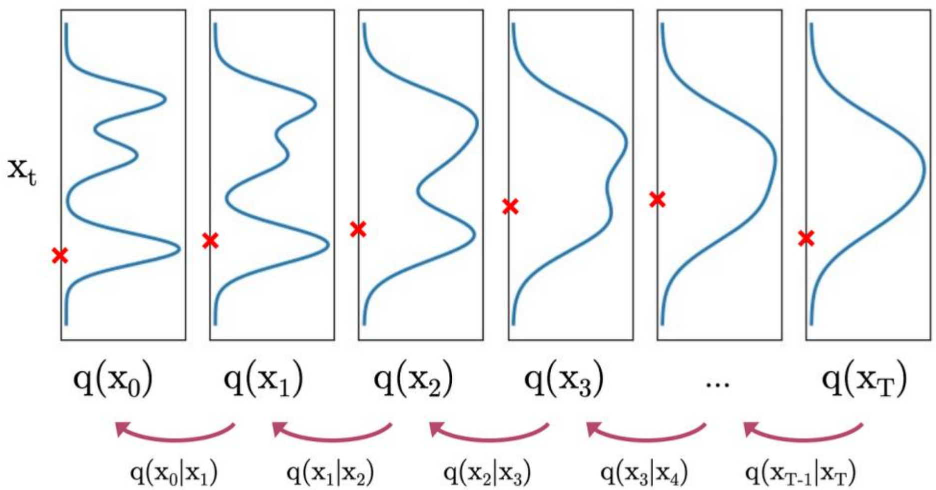
Not very expressive!

Learning distributions

- Mixture of Gaussians

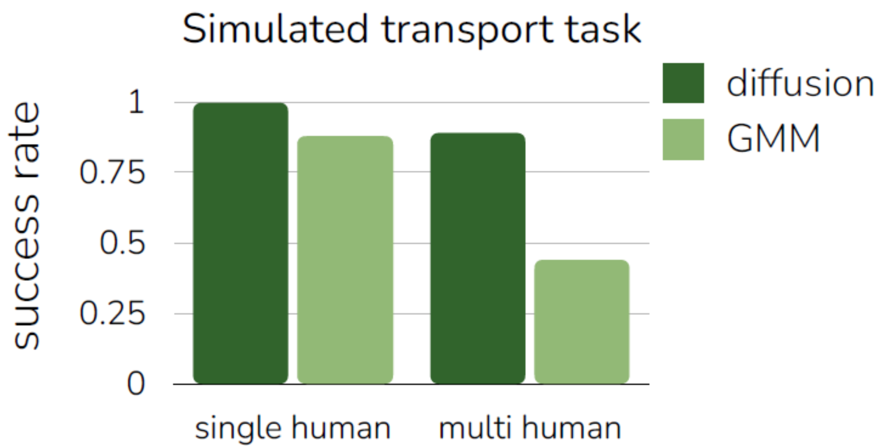


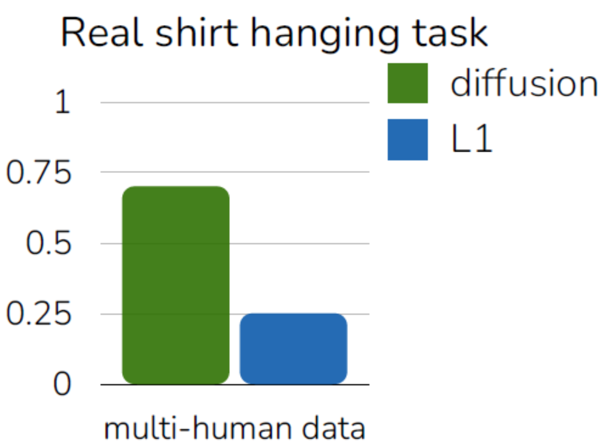
- Diffusion

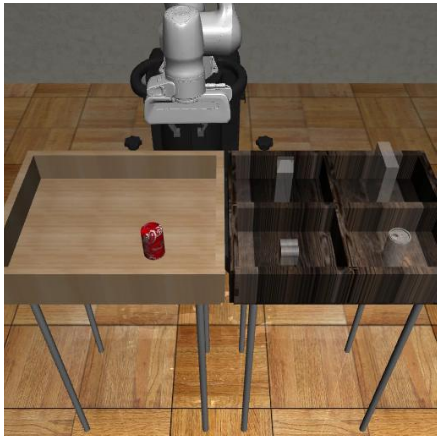


Learning distributions

> Deterministic policy vs distribution policy

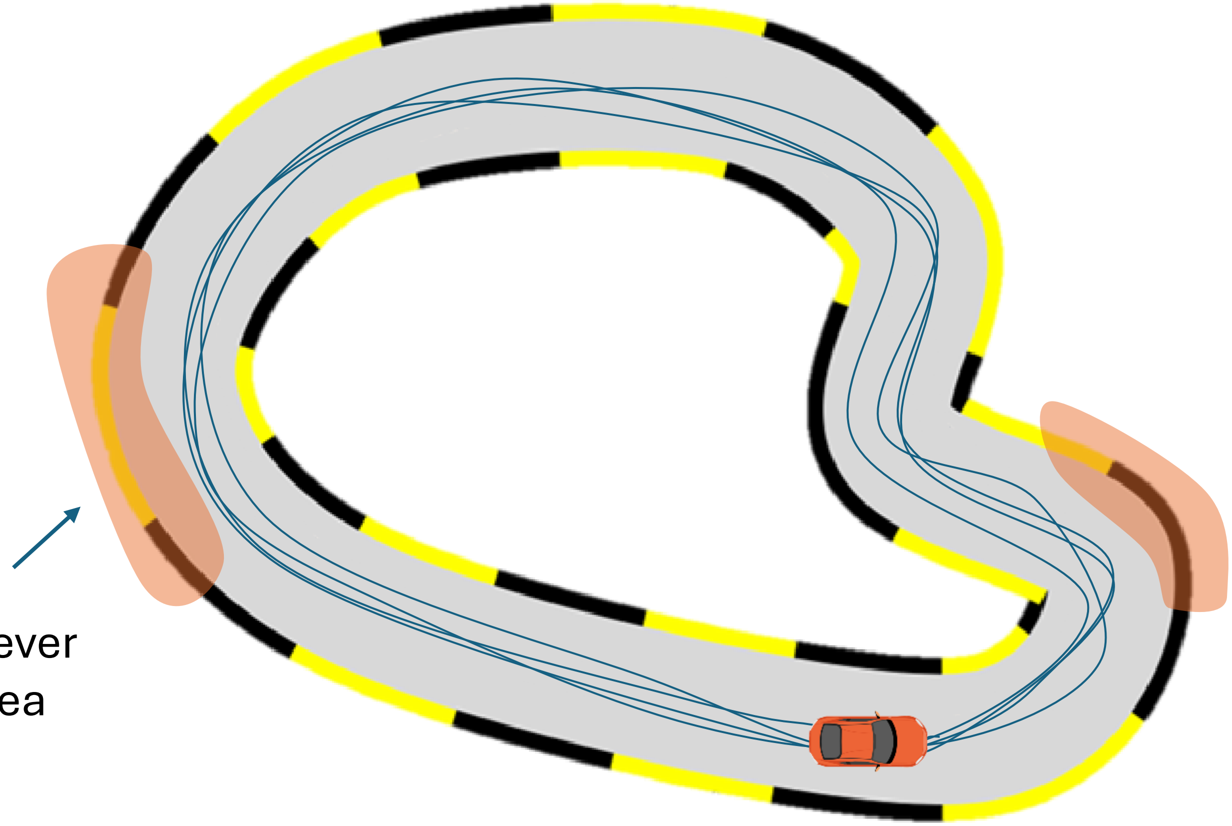






Collect more data

- Be smart about how we collect our data
  - Intentionally add mistakes and corrections



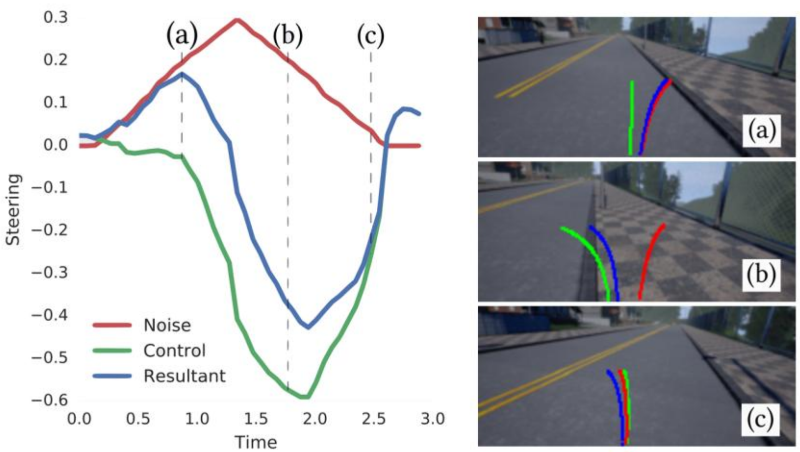
Expert will never reach this area

Collect more data

> Be smart about how we collect our data
  - Intentionally add mistakes and corrections

End-to-end Driving via Conditional Imitation Learning

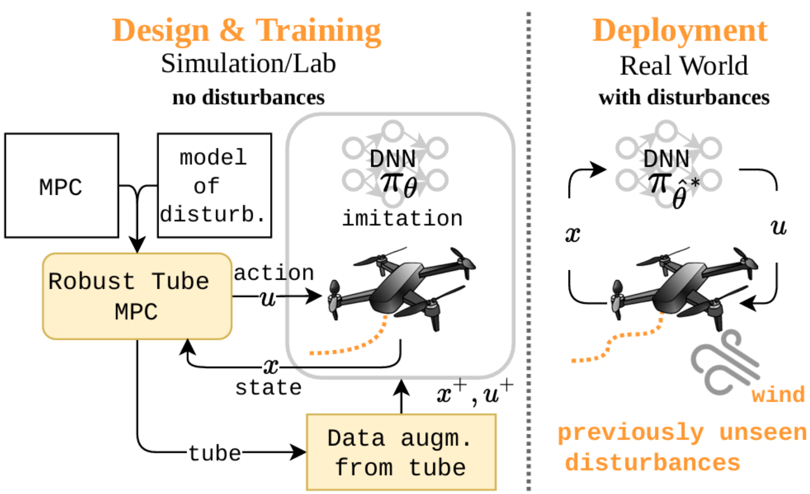
Felipe Codevilla[1,2] Matthias Müller[1,3] Antonio López[2] Vladlen Koltun[1] Alexey Dosovitskiy[1]



perturb the driver's steering angle

Demonstration-Efficient Guided Policy Search via Imitation of Robust Tube MPC

Andrea Tagliabue, Dong-Ki Kim, Michael Everett, Jonathan P. How

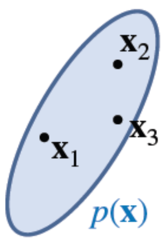


augment an MPC expert using a tube

Collect more data

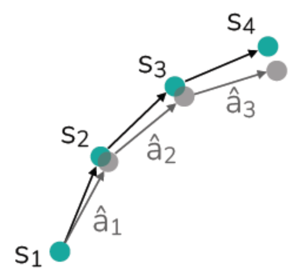
> Compounding errors

Supervised learning



Inputs independent of predicted labels $\hat{\mathbf{y}}$

Supervised learning of behavior



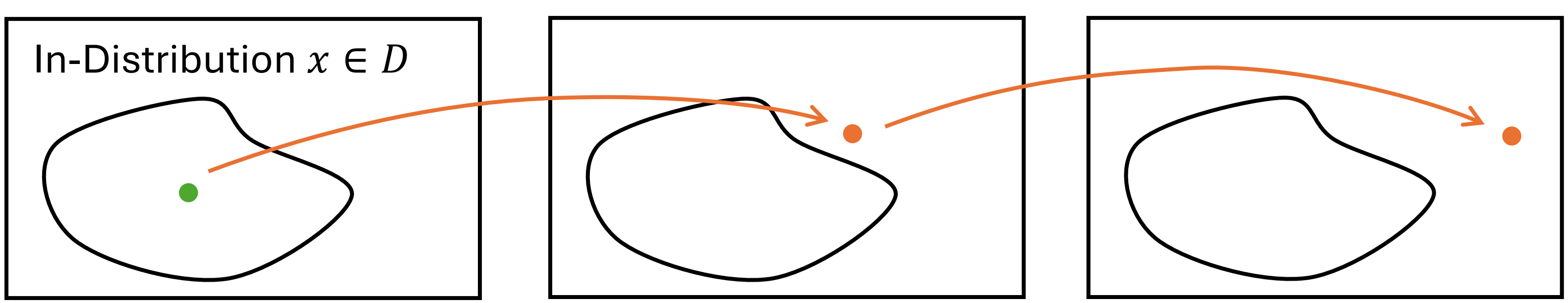
Predicted actions affect next state.

Errors can lead to drift away from the data distribution!

Errors can then compound!

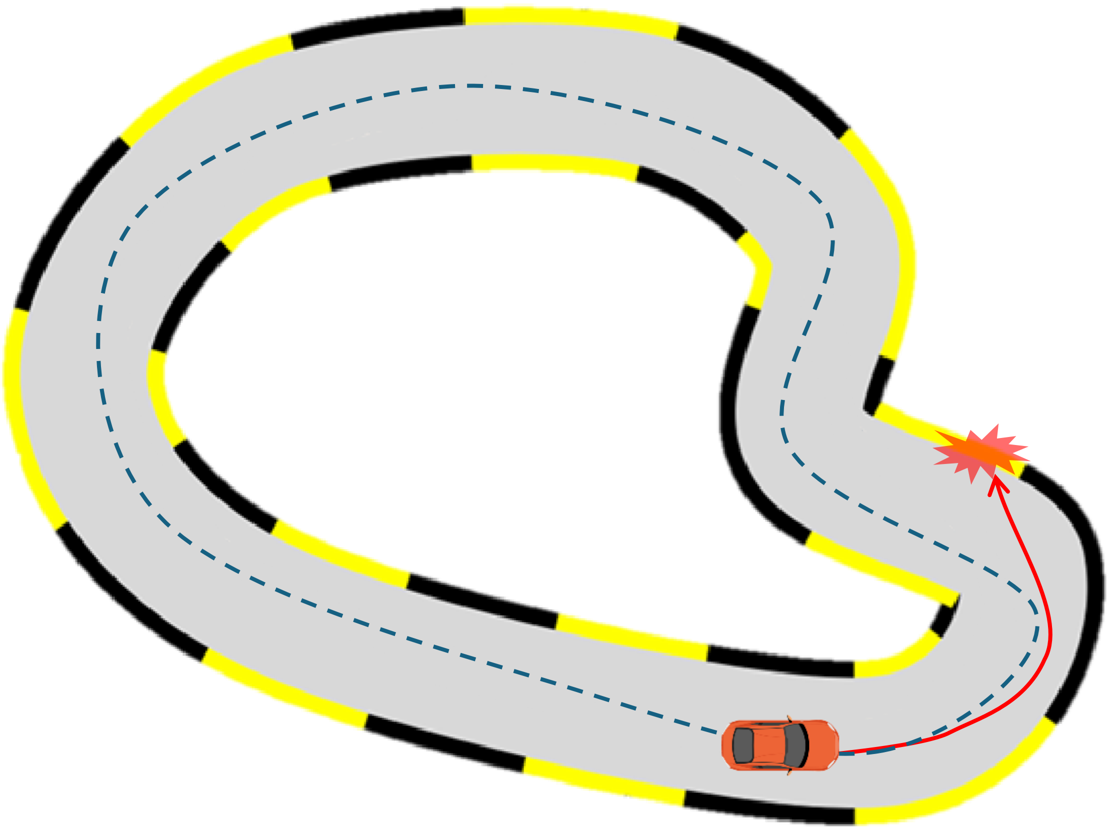
Data space $x \in R^n$

Prediction must be always in distribution



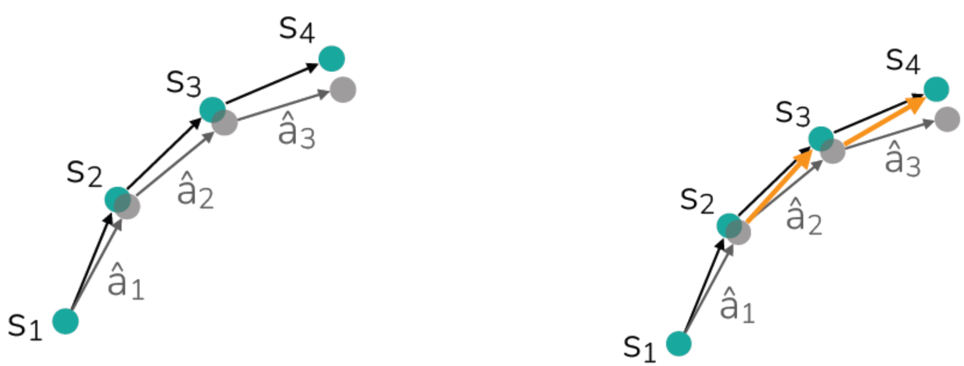
Collect more data

> Trained model might fail



DAgger

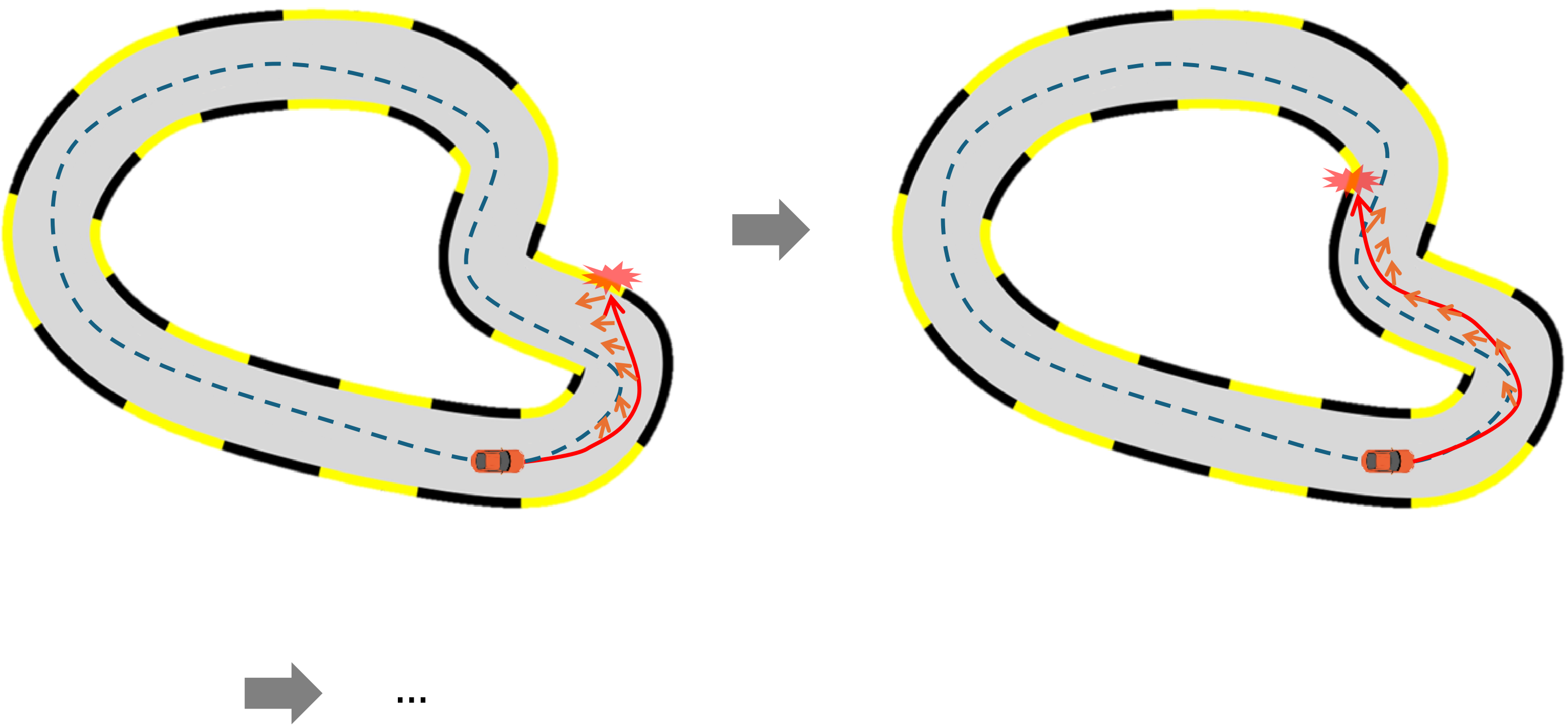
- Trained model might fail
- DAgger: relabel the action for the failed case
  - Train $\pi_\theta(a_t|o_t)$ from human data $D = \{o_1, a_1, \ldots, o_n, a_n\}$
  - Run $\pi_\theta(a_t|o_t)$ to get dataset $D_\pi = \{o_1, \ldots, o_m\}$
  - Ask human to label D_π with action a_t
  - Aggregate: $D \leftarrow D \cup D_\pi$



  - Can be challenging to query expert when agent has control

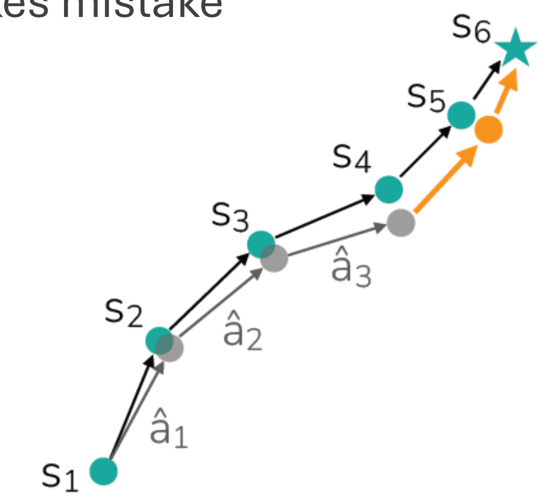
DAgger

- DAgger: dataset aggregation
  - iterate the relabeling until the policy converges



DAgger

- Human gated DAgger
  - Expert intervenes at time t when policy makes mistake
  - Expert provides partial demonstration
  - Aggregate new demos with existing data



  - (much) more practical interface for providing corrections

How to improve more?

- Use good representation
  - Low dimensional features

- Use history
  - POMDP: partially observable, history will tell you the true state
  - Might help for multi modality

IL summary

- Advantages of IL over RL
  - Imitation learning is usually much more sample efficient than RL
  - RL can be challenging when rewards are sparse
  - Designing a reward function can be complex

What are we doing with IL nowadays?

- Tesla humanoid robot