Supervised Learning Recap

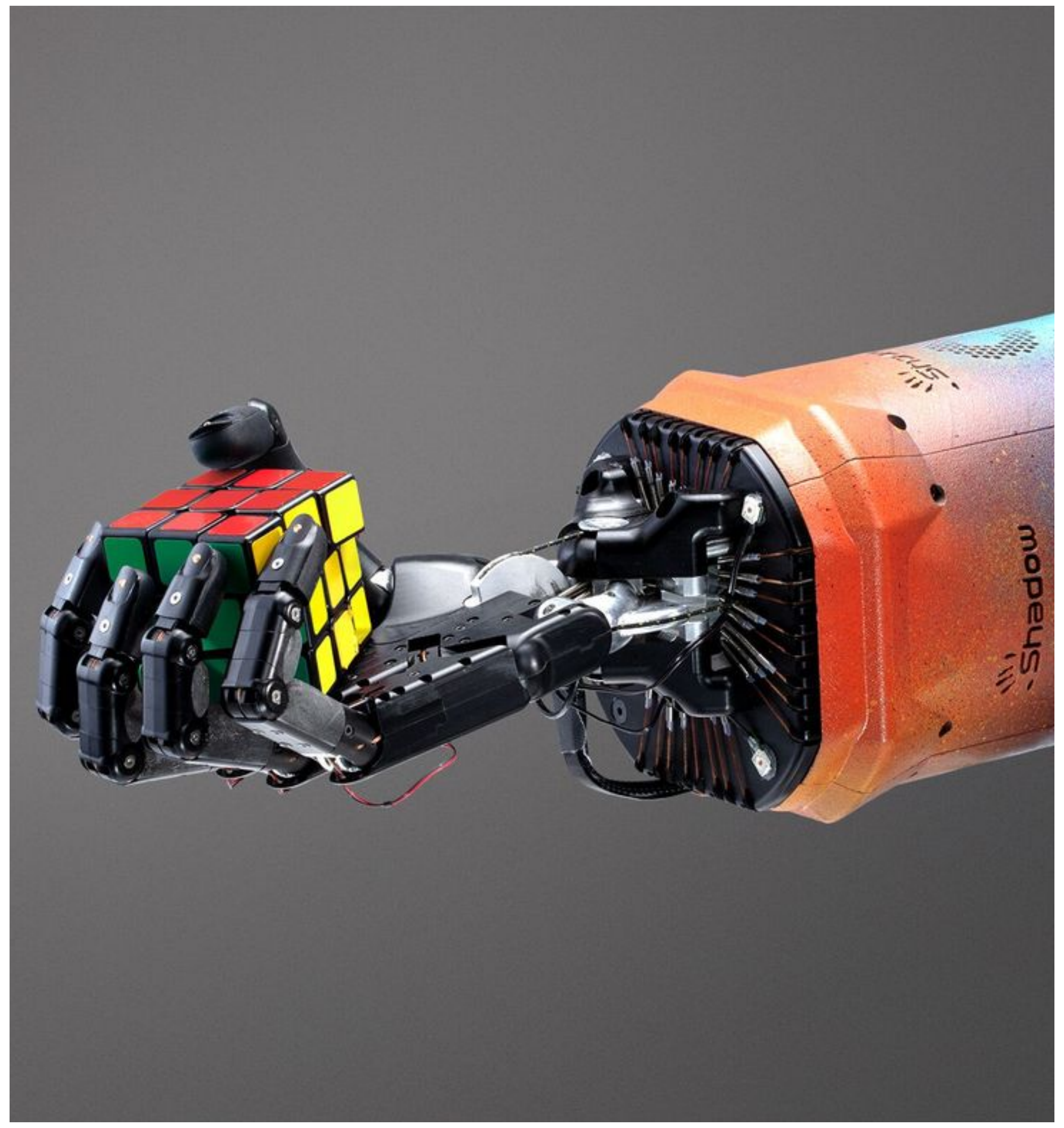
Recap:

So far, three learning algorithms:
TD learning, Q learning, and model-based RL

Limitation: they only work for small MDPs with discrete states and actions

Recap:

Real world problems often have continuous state or extremely large number of states



Cannot hope to **enumerate** all possible state-actions in reality…

Starting from today:

Making RL work for large-scale MDPs with the help from supervised learning
(e.g, Deep Learning)

Outline

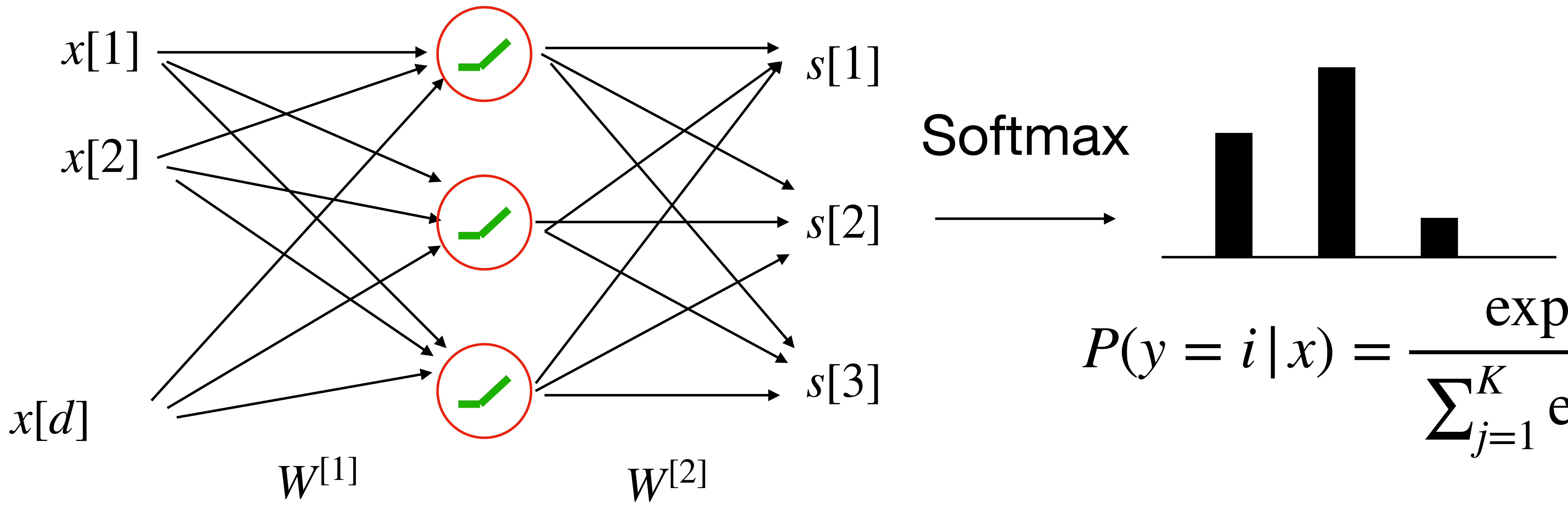
Recap supervised learning

Tutorial on PyTorch and Gym

Multi-class classification

Input: $\mathcal{D} = \{x, y\}, x \in \mathbb{R}^d, y \in \{1,2,\ldots,K\}$

Goal: learn the distribution over labels $P(\cdot \mid x)$



$$P(y = i \mid x) = \frac{\exp(s[i])}{\sum_{j=1}^{K} \exp(s[j])}$$

$$P_\theta(\cdot \mid x) = \text{softmax}\left(W^{[2]}\text{ReLu}(W^{[1]}x)\right)$$

Multi-class classification

Input: $\mathscr{D} = \{x, y\}, x \in \mathbb{R}^d, y \in \{1,2,\ldots,K\}$

Goal: learn the distribution over labels $P(\cdot \mid x)$

Loss function:

Negative log-likelihood

$$\ell(\theta) = \frac{1}{N}\sum_{i=1}^{N} -\ln P_\theta(y^i \mid x^i)$$

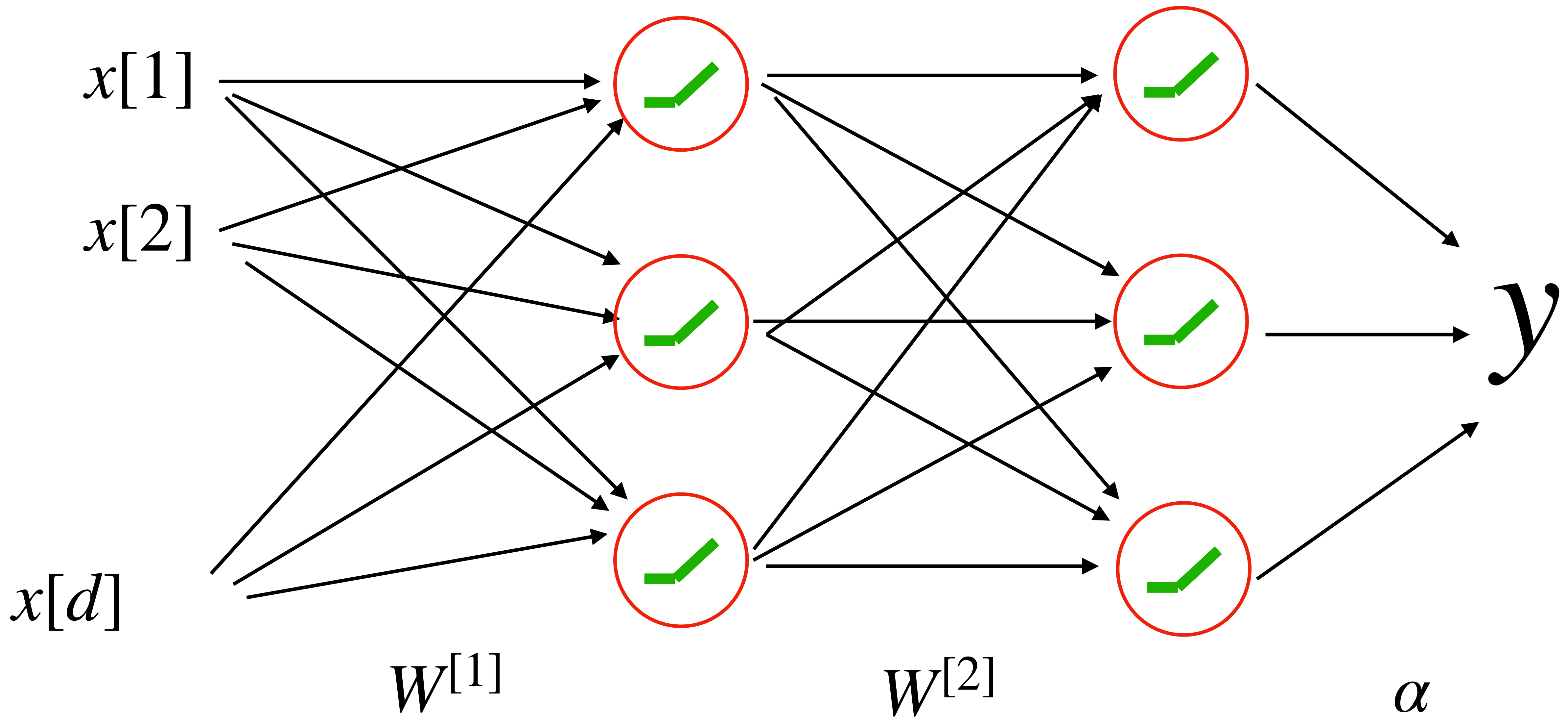
$$\hat{\theta} = \arg\min_{\theta} \ell(\theta)$$

Maximize the likelihood of labels given features

Regression

Input: $\mathscr{D} = \{x, y\}, x \in \mathbb{R}^d, y \in \mathbb{R}, x \sim p, y \sim p(\,.\,|\,x)$

Goal: learn the **Bayes optimal** $\mathbb{E}[y\,|\,x]$



$$y = \alpha^\top \text{ReLU}\left(W^{[2]}\text{ReLU}\left(W^{[1]}x\right)\right)$$

Regression

Input: $\mathscr{D} = \{x, y\}, x \in \mathbb{R}^d, y \in \mathbb{R}$

Goal: given x, learn the **Bayes optimal** $\mathbb{E}[y \mid x]$

Loss function:

Mean square error (MSE)

$$\ell(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(f_\theta(x) - y\right)^2$$

$$\hat{\theta} = \arg\min_{\theta} \ell(\theta)$$

Minimize the mean squared error

What we can hope from supervised learning?

We expect the learned regressor/classifier do well **under the same distribution** where training data is sampled

e.g., for regression, under cerntain assumptions

$$\mathbb{E}_{x\sim p}\left(f_{\hat{\theta}}(x) - \mathbb{E}[y \mid x]\right)^2 \to 0, \text{ as } N \to \infty$$

Dist where training data is sampled

Generalization

Supervised learning exhibits generalization ability, as long as test samples are sampled from the same training dist

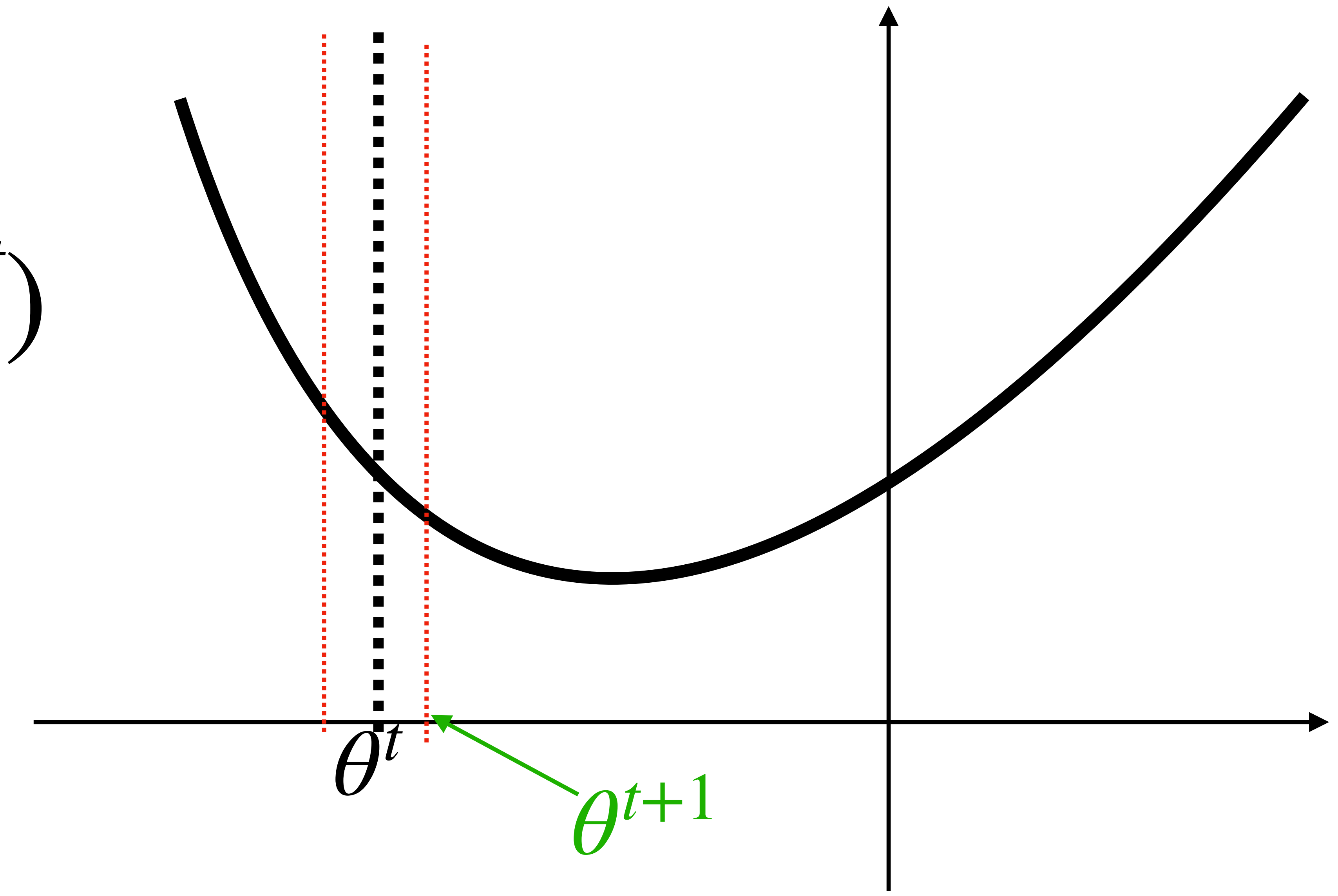
e.g., Classifer trained on large-scale cat / dog images can classifier **unseen** cat or dog images

Optimization

We focus on first-order optimization techiqune: (stochatsic) gradient descent

$$\ell(\theta) = \frac{1}{N}\sum_{i=1}^{N} - \ln P_\theta(y^i \mid x^i)$$

GD: $\theta^{t+1} = \theta^t - \eta \nabla_\theta \ell(\theta^t)$



Optimization

We focus on first-order optimization techiqune: (stochatsic) gradient descent

$$\ell(\theta) = \frac{1}{N} \sum_{i=1}^{N} - \ln P_\theta(y^i \mid x^i)$$

SGD: $$\theta^{t+1} = \theta^t - \eta \widetilde{\nabla_\theta \ell(\theta^t)}$$

Q: how to get this unbiased estimate of the gradient?

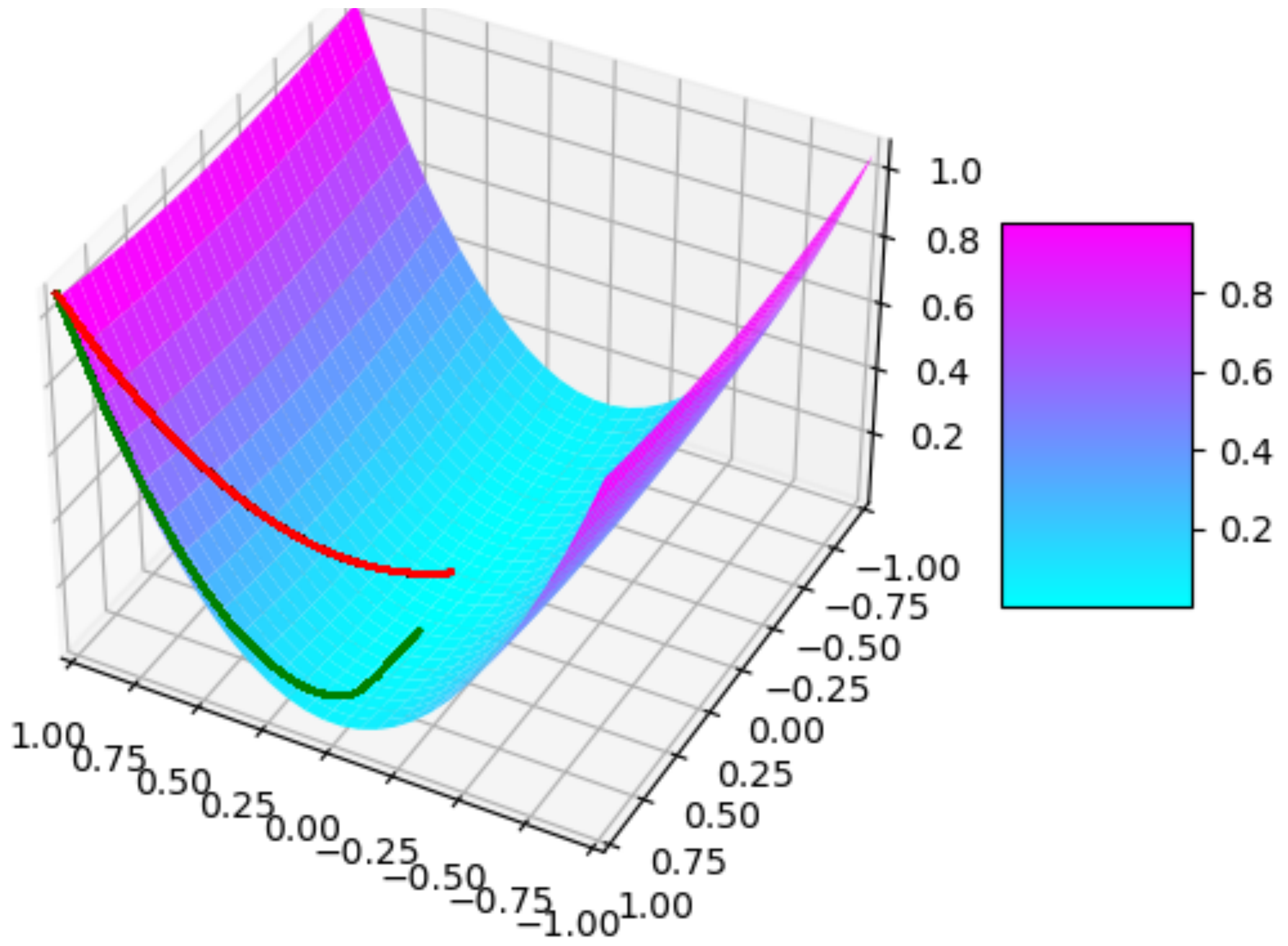
Optimization

Often we use adaptive gradient methods such as Adagrad or Adam:

In high level, adaptively set learning rates for different coordinates and time

Visualization of AdaGrad VS GD

$$\ell(w) = w[1]^2 + 0.01w[2]^2$$



AdaGrad can make good progress on all axis

We often use Adam (Adagrad + momentum) in practice

Outline

Recap supervised learning

Tutorial on PyTorch and Gym