9.5.3 - Coarse Coding

Jan 25, 2022

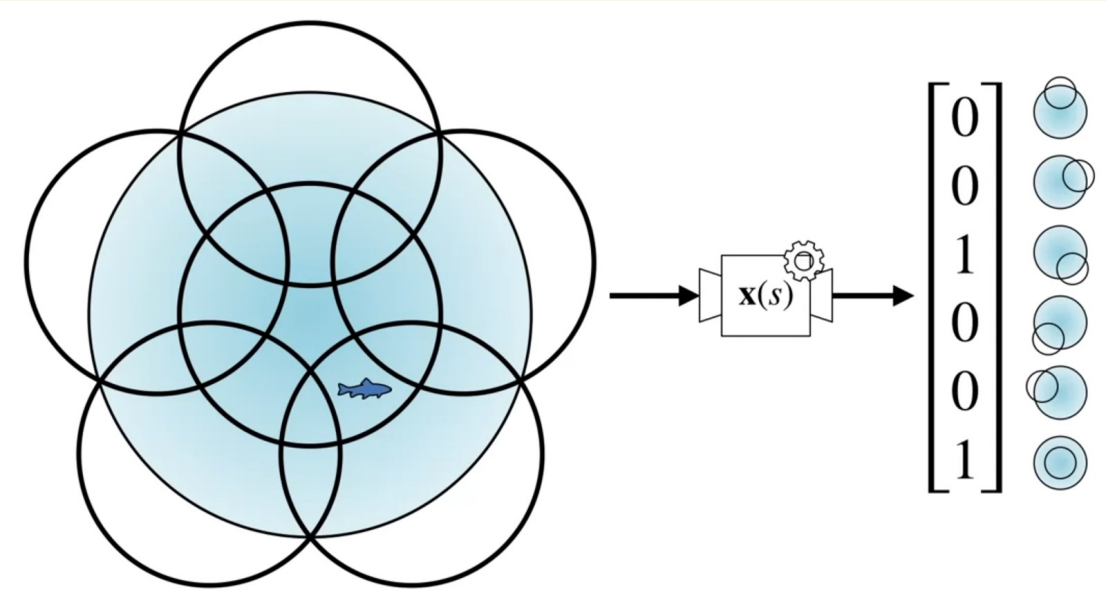
e.g. Continuous 2D space, circles represent features in state space.

↳ If state is inside a circle, the feature has value 1 and is present.

↳ Otherwise feature is 0 and absent.

Representing a state w/ overlapping features this way is coarse coding.

↳ i.e. Generalization between different states depends on the number of their features whose receptive fields overlap.

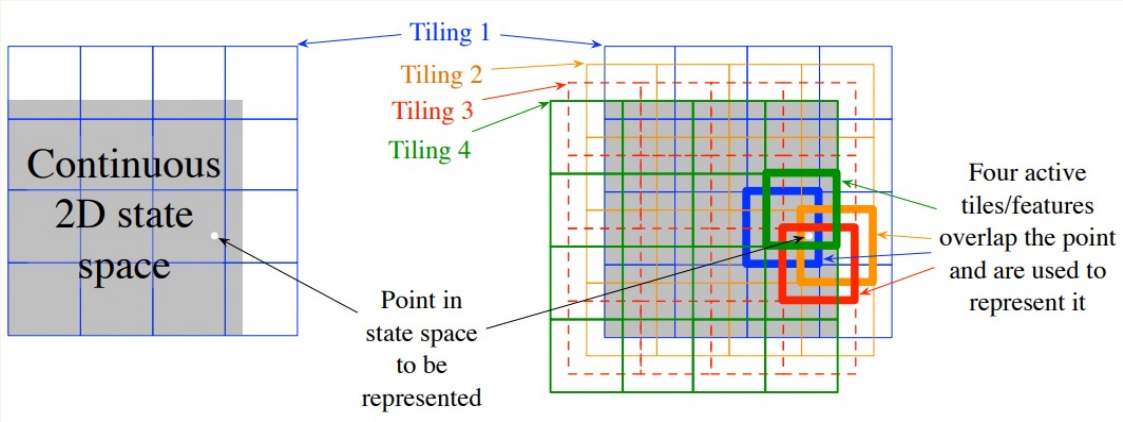


9.5.4 - Tile Coding:

Tile coding is a form of coarse coding for non-2D, multi-dimensional continuous spaces.

↳ Receptive fields of features are grouped into partitions of the state space.

↳ Each partition is a tiling, and each element of the partition is a tile.

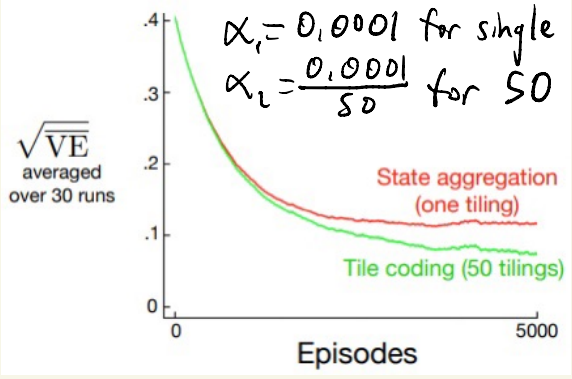


⊗ Total # features present is always the same as the # tilings.

↳ Allows α to be chosen easily and intuitively.

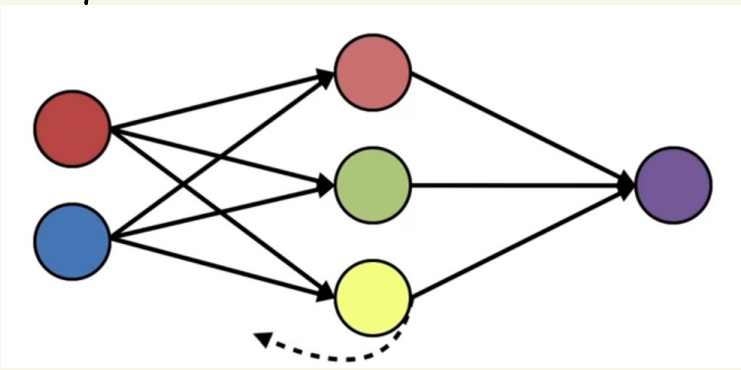
e.g. $\alpha = \frac{1}{n}$ → one-trial learning

n = # tilings



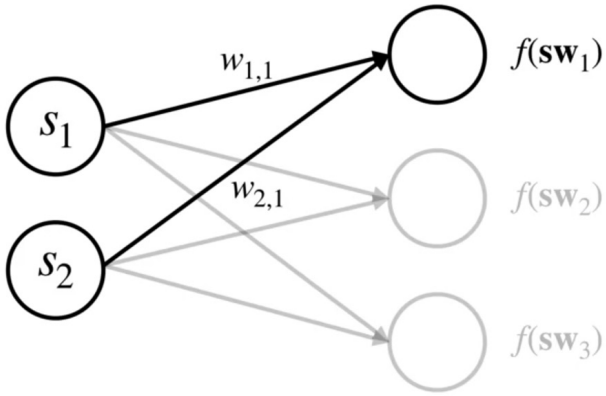
- Tile coding achieves better discrimination than state aggregation due to the small intersections created by overlapping tiles.

Simple Neural Network:



Input Layer Hidden Layer Output Layer

Implementation:



$$\mathbf{s} = [s_1, s_2]$$

$$\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \mathbf{w}_3] = \begin{bmatrix} w_{1,1} & w_{2,1} & w_{3,1} \\ w_{1,2} & w_{2,2} & w_{3,2} \end{bmatrix}$$

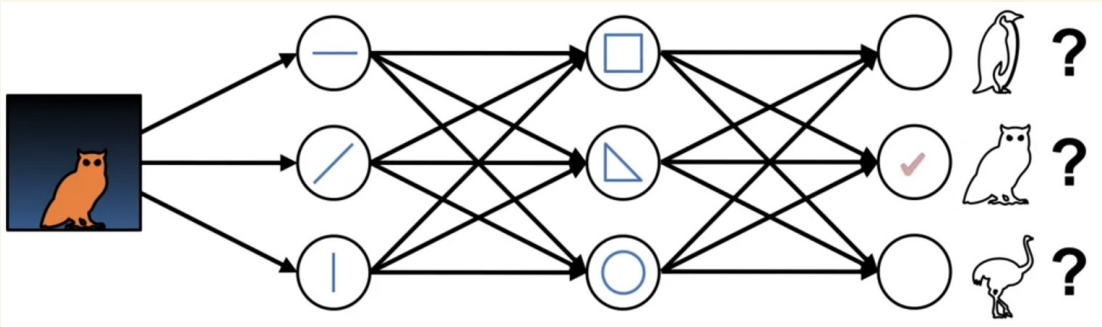
$$\text{outputs} = f(\mathbf{s}\mathbf{W})$$

$\mathbf{s} = [s_1, s_2]$ $\mathbf{w}_1 = [w_{1,1}, w_{2,1}]^T$

Neural Networks Utilize Compositional Features!

Each layer has a different level of abstraction.

↳ Deeper networks facilitate better composition and abstraction.



Deriving Gradient Descent for NN's!

$$\frac{\partial L(\hat{\mathbf{y}}_k, \mathbf{y}_k)}{\partial \mathbf{B}_{jk}} = \frac{\partial L(\hat{\mathbf{y}}_k, \mathbf{y}_k)}{\partial \hat{\mathbf{y}}_k} \frac{\partial \hat{\mathbf{y}}_k}{\partial \mathbf{B}_{jk}}$$

$$= \frac{\partial L(\hat{\mathbf{y}}_k, \mathbf{y}_k)}{\partial \hat{\mathbf{y}}_k} \frac{\partial f_{\mathbf{B}}(\theta_k)}{\partial \theta_k} \frac{\partial \theta_k}{\partial \mathbf{B}_{jk}}$$

$$= \frac{\partial L(\hat{\mathbf{y}}_k, \mathbf{y}_k)}{\partial \hat{\mathbf{y}}_k} \frac{\partial f_{\mathbf{B}}(\theta_k)}{\partial \theta_k} \mathbf{x}_j \qquad \delta_k^{\mathbf{B}} = \frac{\partial L(\hat{\mathbf{y}}_k, \mathbf{y}_k)}{\partial \hat{\mathbf{y}}_k} \frac{\partial f_{\mathbf{B}}(\theta_k)}{\partial \theta_k}$$

$$= \delta_k^{\mathbf{B}} \mathbf{x}_j$$

$$\frac{\partial L(\hat{\mathbf{y}}_k, \mathbf{y}_k)}{\partial \mathbf{A}_{ij}} = \delta_k^{\mathbf{B}} \mathbf{B}_{jk} \frac{\partial f_{\mathbf{A}}(\psi_j)}{\partial \psi_j} \mathbf{s}_i \qquad \delta_j^{\mathbf{A}} = \left(\mathbf{B}_{jk} \delta_k^{\mathbf{B}}\right) \frac{\partial f_{\mathbf{A}}(\psi_j)}{\partial \psi_j}$$

$$= \delta_j^{\mathbf{A}} \mathbf{s}_i$$

$\psi \equiv s\vec{A}$

$\vec{x} \equiv f_{\vec{A}}(\psi)$

$\theta \equiv \vec{x}\vec{B}$

$\hat{y} \equiv f_{\vec{B}}(\theta)$

$\frac{\partial}{\partial} s \leftarrow A \leftarrow x \leftarrow B \leftarrow \hat{y}$

$$\frac{\partial L(\hat{\mathbf{y}}_k, \mathbf{y}_k)}{\partial \mathbf{A}_{ij}} = \delta_j^{\mathbf{A}} \mathbf{s}_i$$

$$\frac{\partial L(\hat{\mathbf{y}}_k, \mathbf{y}_k)}{\partial \mathbf{B}_{jk}} = \delta_k^{\mathbf{B}} \mathbf{x}_j$$

Pseudocode:

for each (s, y) **in** D**:**

$$\delta_k^{\mathbf{B}} = \frac{\partial L(\hat{\mathbf{y}}_k, \mathbf{y}_k)}{\partial \hat{\mathbf{y}}_k} \frac{\partial f_{\mathbf{B}}(\theta_k)}{\partial \theta_k}$$

$$\nabla_{\mathbf{B}}^{jk} = \delta_k^{\mathbf{B}} \mathbf{x}_j$$

$$\mathbf{B} = \mathbf{B} - \alpha_{\mathbf{B}} \nabla_{\mathbf{B}}$$

$$\delta_j^{\mathbf{A}} = \left(\mathbf{B}_{jk} \delta_k^{\mathbf{B}}\right) \frac{\partial f_{\mathbf{A}}(\psi_j)}{\partial \psi_j}$$

$$\nabla_{\mathbf{A}}^{ij} = \delta_j^{\mathbf{A}} \mathbf{s}_i$$

$$\mathbf{A} = \mathbf{A} - \alpha_{\mathbf{A}} \nabla_{\mathbf{A}}$$

Optimization Strategies for NN's:

Weight Initialization:

$\vec{w}_{init} \sim \frac{\mathcal{N}(0, 1)}{\sqrt{n_{inputs}}}$ → initialize from normal distribution

→ scale w/ increasing nodes

Momentum:

$$\vec{w}_{t+1} \leftarrow \vec{w}_t - \alpha \nabla_{\vec{w}} L(\vec{w}_t) + \lambda M_t$$

$$M_{t+1} \leftarrow \lambda M_t - \alpha \nabla_{\vec{w}} L$$