Title of Presentation

This is a more detailed subtitle

Robert Moss
Stanford University

mossr@cs.stanford.edu

Title of slide

Example equation and `algorithmblock` with `juliaverbatim` in a slide.[1]

$$\underbrace{V(s)}_{\text{state value}} \leftarrow \max_a \overbrace{\left(\underbrace{R(s,a)}_{\text{reward}} + \underbrace{\gamma V(s')}_{\substack{\text{discounted value}\\\text{of next state}}}\right)}^{\text{estimate of optimal discounted value}}$$

```julia
# Julia function
V(s) = max(a->R(s,a) + γ*V(s′), A)
```

[1] Mykel J. Kochenderfer and Tim A. Wheeler. *Algorithms for Optimization*. MIT Press, 2019.

Julia console

Example Julia code executing at LaTeX compilation time.

```
julia> using LinearAlgebra

julia> A = Matrix{Int}(I, 3, 3)
3×3 Array{Int64,2}:
 1  0  0
 0  1  0
 0  0  1
```

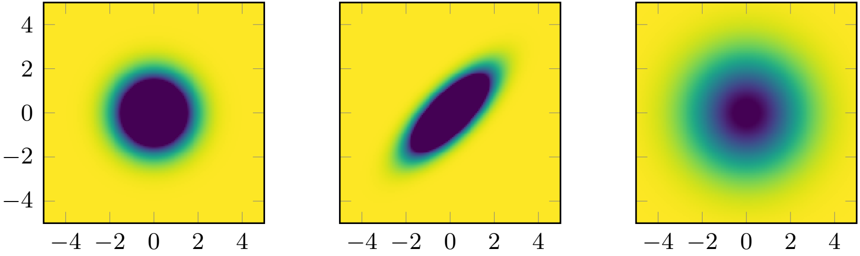
Plotting

Plot using `PGFPlots.jl`[2] directly in the TeX file.

$$\boldsymbol{\mu} = [0, 0] \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\mu} = [0, 0] \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix}$$

$$\boldsymbol{\mu} = [0, 0] \quad \boldsymbol{\Sigma} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$



[2] https://github.com/JuliaTeX/PGFPlots.jl

TikZ figures

For a neural network with one hidden layer, the intermediate hidden units are $h_j = \sigma(\mathbf{v}_j \cdot \phi(x))$ where $\sigma(z) = (1 + e^{-z})^{-1}$, producing output score $= \mathbf{w} \cdot \mathbf{h}$:

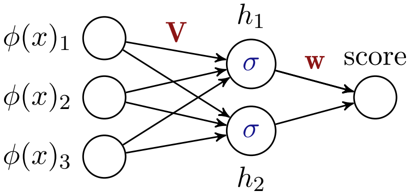


Figure: A *one-layer neural network* with $\phi(x) \in \mathbb{R}^3$ and $\mathbf{h} \in \mathbb{R}^2$ using the logistic activation function σ.

```
σ(z) = 1/(1 + exp(-z)) # sigmoid

function neural_network(x, 𝐕, 𝐰, ϕ, g::Function=σ)
    𝐡 = map(𝐯ⱼ -> g(𝐯ⱼ ⋅ ϕ(x)), 𝐕)
    𝐰 ⋅ 𝐡
end
```

Algorithms

- Monte Carlo tree search (MCTS) is an anytime algorithm that uses rollouts of a random policy to estimate the value of each state-action node in the tree.[3]
- There are four main stages in each simulation: *selection*, *expansion*, *rollout* (or *simulation*), and *backpropagation*.
- The tree $\mathcal{T}$ is iteratively expanded and the policy improves over time as the algorithm balances exploration with exploitation of the state and action spaces.

Algorithm 1 Top-level Monte Carlo tree search algorithm.

```
function MonteCarloTreeSearch(s, d)
    loop
        Simulate(s, d)
    return argmax_{ā∈A(s)} Q(s, ā)
```

Algorithm 2 Monte Carlo tree search simulation.

```
function Simulate(s, d)
    if d = 0
        return 0
    if s ∉ T
        T ← T ∪ {s}
        N(s) ← N_0(s)
        return Rollout(s, d)
    N(s) ← N(s) + 1
    ā ← SelectAction(s)                          ▷ selection
    (s′, r) ← DeterministicStep(s, ā)            ▷ expansion
    q ← r + γ Simulate(s′, d − 1)                ▷ simulation/rollout
    N(s, ā) ← N(s, ā) + 1
    Q(s, ā) ← Q(s, ā) + (q − Q(s, ā)) / N(s, ā)  ▷ backpropagation
    return q
```

[3] Rémi Coulom. "Efficient selectivity and backup operators in Monte-Carlo tree search". In: *International Conference on Computers and Games*. Springer. 2006, pp. 72–83.

Definition blocks

Example use of a `definitionblock` environment with accompanying Julia code.

Definition: score. The score on an example (x, y) is $\mathbf{w} \cdot \phi(x)$, how *confident* we are in predicting $+1$. Score is a weighted combination of features:

$$\mathbf{w} \cdot \phi(x) = \sum_{j=1}^{d} w_j \phi(x)_j$$

```julia
score(x, w, ϕ) = w⋅ϕ(x)
```

Tables

Example table with footnotes.

Table: Algorithm Hyperparameters

Hyperparameter	**Value**
episodes *	5000
maximum tree depth d_{max} (i.e. number of waypoints) *	12
rollout depth d †	12
exploration constant c	10
progressive widening k	10
progressive widening α	0.3

* Used by all algorithms.
† Used by MCTS and direct Monte Carlo.

Example: bullet points

- Bullet point
  - Sub-bullet point
- Another bullet point
  - Another sub-bullet point
    - Another level

References

Coulom, Rémi. "Efficient selectivity and backup operators in Monte-Carlo tree search". In: *International Conference on Computers and Games*. Springer. 2006, pp. 72–83.

Kochenderfer, Mykel J. and Tim A. Wheeler. *Algorithms for Optimization*. MIT Press, 2019.