



1. State Aggregation và mối quan hệ với Coarse Coding

1.1. State Aggregation (Phân nhóm trạng thái)

- **Khái niệm:** State Aggregation (tổng quát: tile grouping) là kỹ thuật đơn giản để giảm số lượng trạng thái cần lưu/truy cập khi xấp xỉ hàm giá trị. Ý tưởng: chia tập trạng thái S thành một số nhóm (aggregates) rời rạc, sao cho mọi trạng thái trong cùng nhóm dùng chung một ước lượng giá trị V . Ví dụ, nếu có 1000 trạng thái nhưng ta chia thành 10 nhóm, chỉ cần lưu 10 tham số thay vì 1000. .
- **Ưu điểm:**
 - Giảm độ phức tạp bộ nhớ và tính toán.
 - Hỗ trợ generalization: khi trải nghiệm một trạng thái trong nhóm, cập nhật ước lượng cũng ảnh hưởng đến cả nhóm, giúp ước lượng cho các trạng thái tương tự.
- **Nhược điểm:**
 - Nếu nhóm chung chứa các trạng thái có giá trị thật khác biệt đáng kể, sẽ gây bias lớn.
 - Thiết kế grouping phụ thuộc kiến thức miền: nếu không hợp lý, underfitting nặng.
- **Mối quan hệ với Linear Approximation:** State Aggregation là trường hợp đặc biệt của linear function approximation với feature one-hot: $\phi_i(s) = 1$ nếu s thuộc nhóm i , ngược lại 0. Khi áp dụng cập nhật TD với feature này, ta nhận lại behavior giống Tabular TD nhưng trên các nhóm thay vì từng trạng thái riêng biệt people.cs.umass.edu.


1.2. Coarse Coding (Mã hóa thô với vùng kích hoạt chồng lấn)

- **Khái niệm:** Khác với state aggregation cố định nhóm rời, Coarse Coding cho phép các vùng (receptive fields) chồng lấn lên nhau, mỗi vùng tương ứng với một feature. Khi trạng thái s rơi vào một số vùng, các feature tương ứng được kích hoạt (thường gán giá trị 1), còn các feature khác là 0. Như vậy, một trạng thái có thể có nhiều feature active đồng thời. .
- **Cách hoạt động:**
 - Xác định một tập các "hình" (shapes) hoặc "vùng" trong không gian trạng thái (ví dụ các hình tròn, hình chữ nhật, v.v.) đại diện cho receptive fields.
 - Mỗi receptive field gán một component trong vector feature. Khi trạng thái s nằm trong vùng đó, feature = 1, nếu không, = 0. Các vùng thường thiết kế sao cho chồng lấn, để trạng thái gần nhau có nhiều feature chung.
 - Ví dụ slide minh họa: vị trí con cá trong ao, với một số hình tròn chồng lên nhau; vị trí con cá được đại diện bởi vector nhị phân, 1 ở những receptive fields đang bao phủ trạng thái đó. Khi cá di chuyển, một số feature thay đổi, nhưng các trạng thái gần nhau thường kích hoạt bộ feature tương tự, giúp generalization. people.cs.umass.edu papers.neurips.cc.
- **Mối quan hệ với State Aggregation:**
 - Nếu state aggregation coi mỗi nhóm là một vùng rời, không chồng lấn, thì coarse coding là mở rộng: các vùng chồng lên, cho phép biểu diễn mịn hơn và linh hoạt hơn.

- Trong state aggregation truyền thống, mỗi trạng thái chỉ thuộc duy nhất một nhóm (feature one-hot); trong coarse coding, mỗi trạng thái có thể thuộc nhiều vùng, do đó feature vector sparse nhưng có nhiều phần tử 1 hơn.
 - **Ưu điểm của Coarse Coding:**
 - **Generalization linh hoạt:** Chồng lấn giúp lan truyền thông tin học từ một trạng thái sang nhiều trạng thái khác mà không cần nhóm cứng.
 - **Khả năng phân biệt (discrimination):** Do có nhiều vùng chồng lấn, model có thể phân biệt tốt hơn các trạng thái gần kề nếu thiết kế receptive fields đủ chi tiết.
 - **Dễ mở rộng không gian đa chiều:** Khi trạng thái nhiều chiều, có thể xây dựng receptive fields dạng lưới, hay các vùng siêu khối chồng lên nhau.
 - **Nhược điểm & rủi ro:**
 - **Thiết kế vùng:** Cần lựa chọn số lượng và hình dạng receptive fields phù hợp; quá ít vùng dẫn đến underfitting, quá nhiều vùng hoặc quá chồng lấn gây overfitting hoặc tốn bộ nhớ.
 - **Chi phí tính toán:** Với số lượng feature large, cập nhật và lưu trọng số có thể tốn kém; tuy nhiên sparse representation (chỉ vài feature active) giúp giảm chi phí.
 - **Bias do grouping không phù hợp:** Nếu receptive fields không khớp với cấu trúc thực của giá trị hàm, sẽ tạo bias. Nghiên cứu adaptive coarse coding đề cập đến việc tự điều chỉnh receptive fields.
 - **Ví dụ cụ thể:**
 - Trong một môi trường 2D liên tục (ví dụ gridworld liên tục), tạo nhiều vùng hình vuông hoặc tròn chồng lấn khắp không gian. Vector feature sparse, dùng linear approximation $\hat{v}(s) = w^T \phi(s)$. Khi agent trải nghiệm trạng thái s , cập nhật w cho các feature active, giúp ảnh hưởng ước lượng cho những trạng thái khác nằm trong cùng các vùng đó.
 - So với state aggregation, coarse coding "mượt" hơn: không phải phân vùng rời, mà overlap giúp model "nhìn" mối quan hệ cục bộ giữa các trạng thái. papers.neurips.cc.
 - **Ứng dụng:**
 - Dùng trong TD learning hoặc Q-learning với linear approximation.
 - Là nền tảng của CMAC (Cerebellar Model Arithmetic Computer) trong neural networks cổ điển.
 - Có thể kết hợp với eligibility traces (TD(λ)) để tăng tốc lan truyền thông tin.
 - **Tài liệu tham khảo:**
 - R. Sutton & A. Barto, *Reinforcement Learning: An Introduction*, Chương về Function Approximation: Coarse Coding;
 - Nghiên cứu "Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding" (Boyan & Moore, NIPS);
 - Phần mềm tile coding (tiles3) trên trang Sutton's incompleteideas.net.
-

2. Discrimination và Generalization ảnh hưởng đến độ chính xác học

Slide nhấn mạnh hai khái niệm này đều quan trọng và cần cân bằng:

- **Discrimination (Khả năng phân biệt chi tiết)**
 - **Định nghĩa:** Khả năng của mô hình để nhận diện và phản ứng khác biệt với các trạng thái hoặc mẫu dữ liệu có khác biệt tinh vi. Trong RL, discrimination thể hiện ở khả năng hàm xấp xỉ (function approximator) phân biệt các trạng thái khác nhau khi giá trị thật có sự khác biệt quan trọng.  .
 - **Tác động đến Learning Accuracy:**
 - Khi fine-grained distinctions là quan trọng (ví dụ các trạng thái gần nhau nhưng giá trị khác nhau nhiều), model cần đủ capacity/discrimination để học. Nếu feature quá "thô" (ví dụ coarse coding quá rộng), model không phân biệt được, dẫn đến underfitting.
 - Tuy nhiên, quá nhiều discrimination (feature quá phức tạp, quá nhiều parameter), nhất là khi dữ liệu trải nghiệm hạn chế hoặc có noise, có thể dẫn đến overfitting: model ghi nhớ chi tiết không đáng tin cậy, giảm khả năng generalize cho trạng thái chưa gặp. machinelearningmastery.com .
- **Generalization (Khả năng tổng quát hóa)**
 - **Định nghĩa:** Khả năng của mô hình khi áp vào trạng thái/chữ hướng mới chưa gặp trong training vẫn cho ước lượng/gợi ý gần đúng giá trị thật. Trong RL, generalization thể hiện ở việc một trải nghiệm trên một trạng thái hỗ trợ ước lượng cho các trạng thái tương tự khác.
 - **Tác động đến Learning Accuracy:**
 - Generalization tốt giúp tiết kiệm sample: không cần trải nghiệm mọi trạng thái, trải nghiệm một số đủ đại diện có thể lan truyền thông tin. Khi không gian trạng thái rất lớn hoặc liên tục, generalization là điều kiện cần để học hiệu quả.
 - Tuy nhiên, quá mức generalization (feature quá đơn giản, grouping quá rộng) dẫn đến underfitting: model bỏ sót các chi tiết quan trọng, ước lượng sai ngay cả trên dữ liệu đã trải nghiệm. exxactcorp.com .
- **Trade-off giữa Discrimination và Generalization**
 - **Bias-Variance Analogy:** Trong ML, discrimination cao tương đương giảm bias (model linh hoạt hơn), nhưng tăng variance (nhạy cảm hơn với noise/training sample cụ thể). Ngược lại, generalization mạnh (feature đơn giản) dẫn đến bias cao (underfitting) nhưng variance thấp. Cần cân bằng để đạt hiệu năng tổng thể tốt. machinelearningmastery.com .
 - **Trong RL:**
 - Thiết kế feature (coarse coding, tile coding, RBF, neural network) quyết định trade-off này. Ví dụ coarse coding với receptive fields rộng có generalization mạnh nhưng discrimination yếu; receptive fields nhỏ/chi tiết hơn tăng discrimination nhưng cần nhiều trải nghiệm để ước lượng chính xác các trọng số.
 - Phân phối trải nghiệm (experience distribution) cũng ảnh hưởng: nếu policy khám phá tốt, có nhiều sample đa dạng, có thể dùng feature chi tiết hơn; nếu dữ liệu giới hạn, nên chọn feature tổng quát hơn để tránh overfitting noise.

- **Ví dụ minh họa:**
 - Trong pond-fish example, nếu receptive fields quá rộng (cover nhiều vị trí khác biệt), model không phân biệt được khi giá trị thay đổi đáng kể theo vị trí; nếu quá nhiều vùng rất nhỏ, model cần trải nghiệm hầu hết vùng để học chính xác, tốn sample.
- **Kỹ thuật hỗ trợ:**
 - **Adaptive feature construction:** Tự động điều chỉnh kích thước hoặc vị trí receptive fields dựa trên dữ liệu thu được.
 - **Tile Coding với nhiều tilings:** kết hợp nhiều lưới (tilings) chồng lên nhau với offset khác nhau để vừa generalization vừa discrimination.
 - **Regularization:** Giảm overfitting khi dùng feature nhiều dimension.
- **Tham khảo thêm về discrimination/generalization trong ML:**
 - Bài viết về bias-variance trade-off; các nghiên cứu fairness: khái niệm discrimination trong nghĩa xã hội và kỹ thuật kiểm soát bias trong dữ liệu; tuy nhiên ở slide này "discrimination" chủ yếu theo nghĩa kỹ thuật: khả năng phân biệt giữa trạng thái. Nếu mở rộng ứng dụng RL vào môi trường thực (ví dụ đề xuất, tuyển dụng), cần thêm khái niệm fairness để tránh discrimination không mong muốn. superwise.ai.

3. Tile Coding (Mã ô chồng)

Tile Coding là kỹ thuật phổ biến để xây dựng feature cho linear methods trên không gian trạng thái liên tục hoặc đa chiều.

3.1. Nguyên lý hoạt động

- **Partitioning (Chia ô):**
 - Xác định một số tilings (lưới) chia không gian trạng thái thành các ô (tiles). Mỗi tiling là một phân vùng rời (non-overlapping) của không gian.
 - Các tilings được dịch (offset) khác nhau để ô chồng lên nhau theo các cách khác biệt. Khi trạng thái s rơi vào một tile nhất định trên mỗi tiling, tile đó được kích hoạt.
- **Feature Representation:**
 - Mỗi tile tương ứng một feature nhị phân. Với n tilings và mỗi tiling chia thành m ô, tổng số feature = $n \times m$. Tuy nhiên tại một trạng thái s , chỉ có đúng n feature active (một trên mỗi tiling), do vậy vector feature rất sparse.
 - Ví dụ: nếu 4 tilings, mỗi tiling 10×10 ô, tổng feature = $4 \times 100 = 400$, nhưng tại mỗi trạng thái chỉ 4 feature = 1, còn lại 0.
- **Chồng lấn & Discrimination/Generalization:**
 - Offset giữa các tilings giúp tăng discrimination: trạng thái gần nhau có thể cùng kích hoạt nhiều feature chung, nhưng có thể khác ở một số tilings, giúp phân biệt tinh vi. Đồng thời generalization: do tile rộng, trạng thái gần nhau thường cùng tile ở nhiều tilings.

- Việc chọn số tilings và kích thước ô quyết định trade-off: nhiều tilings và ô nhỏ tăng discrimination nhưng tốn bộ nhớ/trải nghiệm; ít tilings và ô lớn tăng generalization nhưng bias cao.
- **Ưu điểm:**
 - **Sparse và hiệu quả:** Vector feature sparse, cập nhật weight nhanh (cộng/trừ một số lượng nhỏ components).
 - **Linear computation:** Với linear approximation, ước lượng giá trị $\hat{v}(s) = w^T \phi(s)$ rất dễ tính (cộng weight của các tile active).
 - **Khả năng mở rộng đa chiều:** Dễ áp dụng cho nhiều chiều (dùng lưới nhiều chiều).
 - **Tính đơn giản:** Dễ triển khai, có sẵn thư viện tile coding (e.g., tiles3). techblog.criteo.com .
- **Nhược điểm & giới hạn:**
 - **Thiết kế tham số:** Cần chọn số tilings, kích thước ô, offset hợp lý; thường dựa trên thử nghiệm.
 - **Cao dimension:** Khi số chiều lớn, số ô tăng nhanh theo lũy thừa, dẫn đến feature khổng lồ. Dùng hashing hoặc CMAC để băm tile vào không gian kích thước giới hạn, nhưng có rủi ro collision.
 - **Không tự động học feature:** Khác với deep learning, tile coding là feature thủ công dựa trên domain knowledge; với môi trường phức tạp, có thể không đủ biểu diễn.
- **Ví dụ minh họa:**
 - Trạng thái 2 chiều (x, y) thuộc $[0,1] \times [0,1]$. Chọn 4 tilings: mỗi tiling chia thành lưới 10×10 , offset khác nhau (ví dụ dịch nửa ô theo x hoặc y). Khi agent ở (x, y) , xác định 4 ô active (mỗi tiling). Vector feature length 400, 4 phần tử 1. Khi agent di chuyển, một vài tile đổi, chia sẻ nhiều tile với trạng thái gần.
- **Ứng dụng trong TD Learning:**
 - Thay vì dùng feature one-hot, dùng tile-coded feature $\phi(s)$ để ước lượng $\hat{v}(s) = w^T \phi(s)$. Cập nhật TD: $w \leftarrow w + \alpha \delta_t \phi(s_t)$. Vì sparse, mỗi cập nhật chỉ ảnh hưởng weight của các tile active.
 - Kết hợp với eligibility traces: accumulate traces trên tile, lan truyền thông tin hiệu quả.
- **Tài liệu tham khảo:**
 - Sutton & Barto, *Reinforcement Learning: An Introduction*, Chương 9, phần Tile Coding;
 - Trang "Tile Coding Software" của Sutton's incompleteideas.net incompleteideas.net .
 - Nguồn lịch sử: CMAC của Albus (1975) – tiền thân concept tile coding.
 - Bài viết blog "Tile-Coding: An Efficient Sparse-Coding Method for Real-Valued Data" minh họa trực quan. techblog.criteo.com .

4. Ảnh hưởng của Feature Construction đến Học (Learning Accuracy)

4.1. Vai trò của Feature trong Linear Methods

- Với linear approximation, giá trị $\hat{v}(s) = w^\top \phi(s)$. Feature $\phi(s)$ quyết định khả năng mô hình biểu diễn giá trị hàm.
- **Chất lượng feature** quyết định bias và variance của ước lượng:
 - Feature giàu (rich) cho khả năng biểu diễn phức tạp, giảm bias nhưng tăng variance.
 - Feature nghèo (coarse) cho generalization mạnh, giảm variance nhưng bias cao.
- **Ví dụ:** Coarse Coding với receptive fields rộng (feature nghèo) dẫn đến generalization tốt nhưng không phân biệt đủ; Tile Coding với nhiều tilings tăng capacity, giảm bias nhưng cần nhiều sample.

4.2. Discrimination vs Generalization trong RL

- **Discrimination:** Feature phải đủ nhạy cảm để phân biệt các trạng thái có giá trị khác nhau; nếu không, model không thể học fine distinctions.
- **Generalization:** Feature cần chia sẻ thông tin giữa các trạng thái tương tự; nếu không, mỗi trạng thái riêng lẻ, cần trải nghiệm nhiều để ước lượng.
- **Trade-off:** Lựa chọn feature construction (số vùng, kích thước vùng, số tilings, v.v.) sao cho cân bằng. Với coarse coding, receptive fields chồng lấn vừa phải; với tile coding, số tilings và độ mịn của lưới cân nhắc theo độ phức tạp môi trường và giới hạn sample. [machinelearningmastery.com](https://machinelearningmastery.com/exact-coding-in-reinforcement-learning/)
[exactcorp.com](https://www.exactcorp.com/).
- **Thiết kế adaptive:** Một số nghiên cứu đề xuất tự điều chỉnh kích thước hoặc vị trí vùng dựa trên quãng thời gian agent khám phá, tăng discrimination ở vùng quan trọng, tăng generalization ở vùng ít quan trọng.

4.3. Rủi ro Overfitting/Underfitting

- **Underfitting:** Khi feature quá đơn giản (ví dụ state aggregation quá thô, ít vùng coarse coding hoặc tile coding ít tilings/ô lớn), model không bắt hết variation của giá trị, độ lỗi ước lượng cao ngay cả trên trạng thái đã trải nghiệm.
- **Overfitting:** Khi feature quá phức tạp so với dữ liệu trải nghiệm (ví dụ quá nhiều tilings, receptive fields rất nhỏ) nhưng sample ít hoặc nhiễu cao, model học noise, ước lượng sai ở trạng thái chưa gặp.
- **Mitigation:**
 - Điều chỉnh số lượng feature dựa trên độ đa dạng sample;
 - Regularization (trong linear methods, có thể weight decay);
 - Replay buffer, experience replay để làm giàu trải nghiệm cho tile coding;
 - Kiểm tra performance trên validation set (trong RL, kiểm tra ước lượng giá trị hoặc performance policy) để điều chỉnh feature.

5. Ứng dụng và Ví dụ cụ thể

5.1. Ví dụ Coarse Coding trong RL

- Giả sử môi trường 2D: vị trí agent (x, y) . Tạo 20 receptive fields dưới dạng các hình tròn ngẫu nhiên có bán kính khác nhau, chồng lấn phân bố khắp không gian. Khi agent ở vị trí s , xác định subset các receptive fields chứa s ; feature vector sparse kích hoạt những indices đó. Linear approximation cập nhật weight cho các receptive fields active, lan truyền thông tin sang vùng gần đó.
- So sánh với state aggregation cứng: nếu nhóm cố định theo ô vuông lớn, agent không phân biệt fine distinctions; coarse coding linh hoạt hơn nhưng vẫn tiết kiệm parameter so với tile coding chi tiết.

5.2. Ví dụ Tile Coding trong RL

- Với môi trường CartPole (2D trạng thái: góc trục + vận tốc góc), dùng tile coding:
 - Chọn 8 tilings, mỗi tiling chia biên góc và vận tốc thành grid 8×8 , offset khác nhau. Khi trạng thái $(\theta, \dot{\theta})$, xác định 8 tile active. Feature vector length = $8 \times 8 \times 8 = 512$, sparse với 8 phần tử 1.
 - Dùng linear Q-learning: $Q(s, a) \approx w_a^T \phi(s)$ (mỗi hành động có bộ weight riêng). Cập nhật TD: $w_a \leftarrow w_a + \alpha \delta \phi(s)$.
 - Kết quả: so với biểu diễn liên tục trực tiếp, tile coding giúp khái quát hóa, học nhanh hơn. Nhưng nếu grid quá mịn hoặc quá thô, performance thay đổi.
- Các thư viện tile coding (ví dụ tiles3) hỗ trợ triển khai.

5.3. Sử dụng trong TD learning

- Khi dùng tile-coded feature, TD update sparse, mỗi bước chỉ thay đổi weight tương ứng tile active. Nhờ generalization và sparse update, sample efficiency tốt.
- Kết hợp với eligibility traces: trace gắn với tile, khi reward mới xuất hiện, trace lan truyền tới các tile đã active trước đó, tăng tốc học credit assignment.
- Trong policy gradient hoặc actor-critic, tile coding có thể dùng để xấp xỉ value (critic) hoặc trực tiếp policy parameterization (actor).

6. Tóm tắt và khuyến nghị khi xây dựng feature cho Linear Methods

1. **Xác định mục tiêu và giới hạn:** Tính chất môi trường (độ dimension, continuous/discrete, phân phối trạng thái) và tài nguyên sample/bộ nhớ.
2. **Bắt đầu từ simple:** Có thể thử state aggregation hoặc coarse coding đơn giản để kiểm tra khái quát; nếu không đủ, tăng độ tinh vi của feature.
3. **Chọn coarse coding hay tile coding:**

- Coarse coding phù hợp khi muốn linh hoạt chồng lấn nhưng không cần lưới đều; có thể tự định nghĩa các receptive fields phù hợp domain.
 - Tile coding phù hợp khi trạng thái có thể chia đều thành grid, và muốn sử dụng khả năng phân biệt tinh vi qua nhiều tilings offset.
4. **Thiết lập trade-off discrimination/generalization:** Chọn số receptive fields hoặc số tilings/kích thước ô phù hợp với sample budget và độ phức tạp hàm giá trị. Test qua các cấu hình khác nhau, theo dõi learning curves.
 5. **Sparse representation:** Ưu tiên sparse feature để giảm chi phí tính toán và tránh overfitting. Tile coding và coarse coding thường tạo vector sparse.
 6. **Adaptive adjustments:** Nếu môi trường có vùng quan trọng (frequent visits hoặc có reward đột biến), có thể tập trung thêm receptive fields hoặc tilings nhỏ hơn ở vùng đó để tăng discrimination.
 7. **Regularization và stability:** Với linear methods, weight decay đơn giản có thể hữu ích khi feature dimension lớn; trong RL, điều chỉnh learning rate, eligibility trace parameters để ổn định.