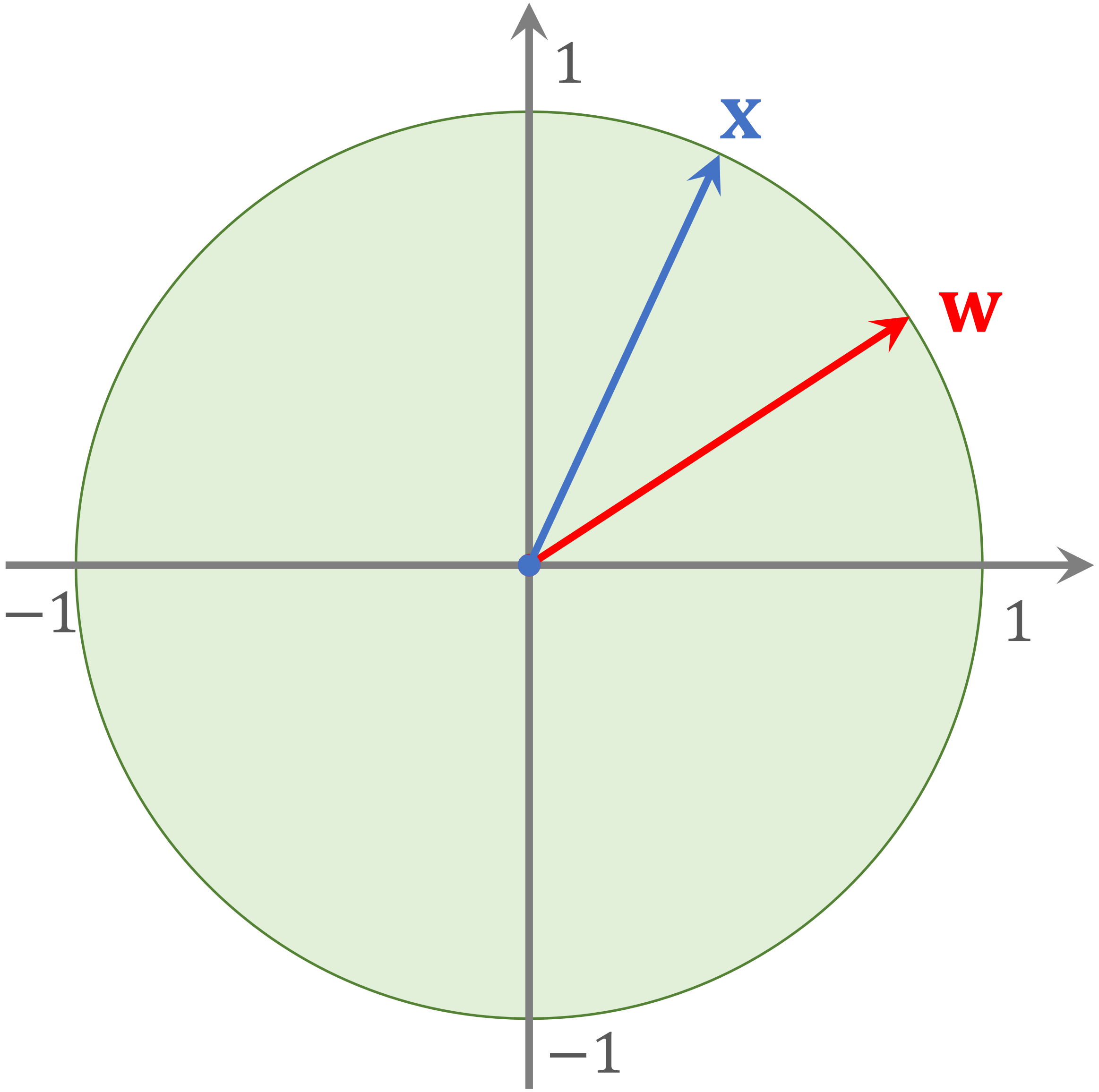
Pretraining and Fine Tuning

Shusen Wang

Preliminary

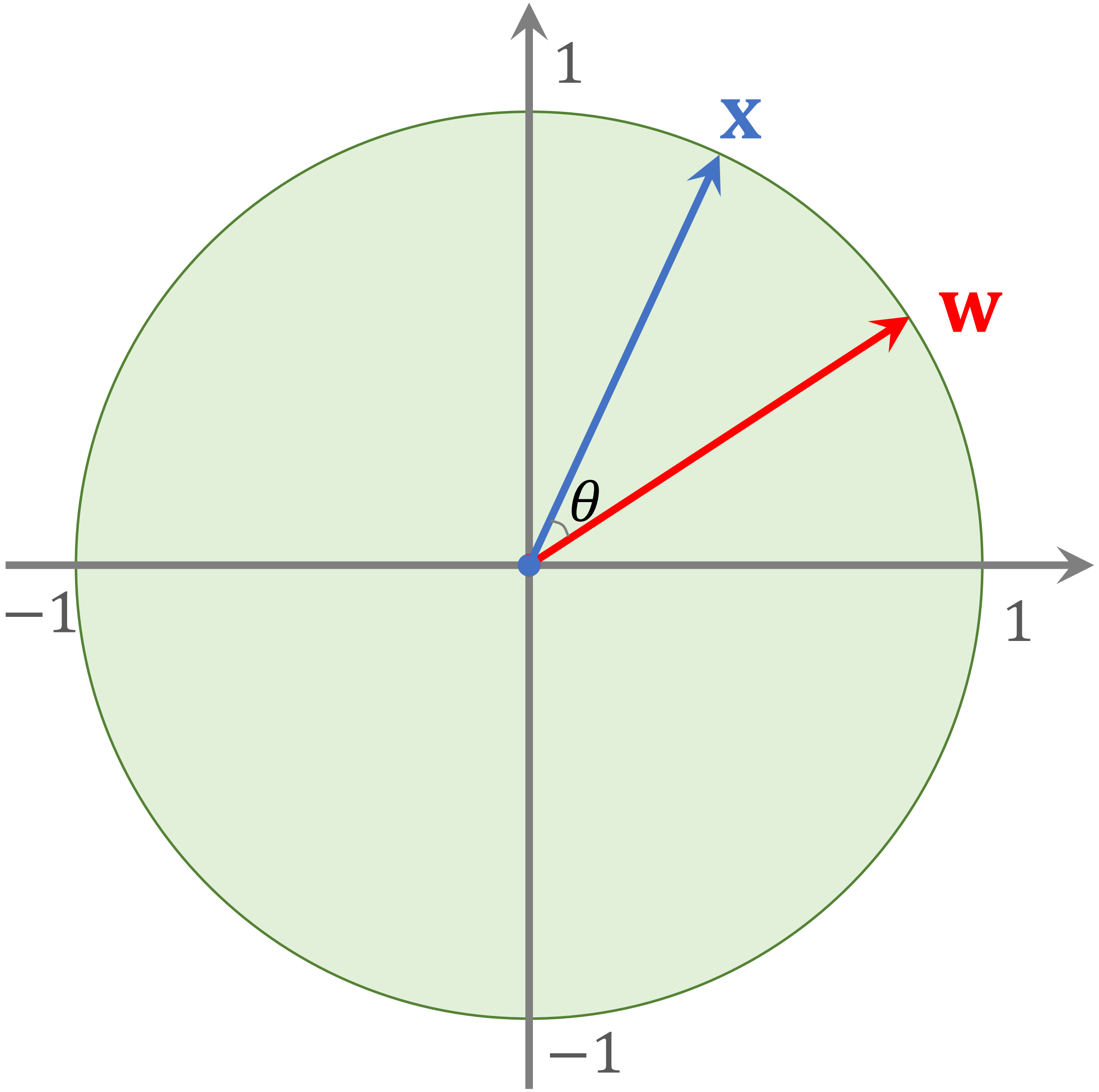
Cosine Similarity



- Assume $\mathbf{x}$ and $\mathbf{w}$ are unit vectors:

  $\left|\left|\mathbf{x}\right|\right|_2 = 1$ and $\left|\left|\mathbf{w}\right|\right|_2 = 1$.

Cosine Similarity



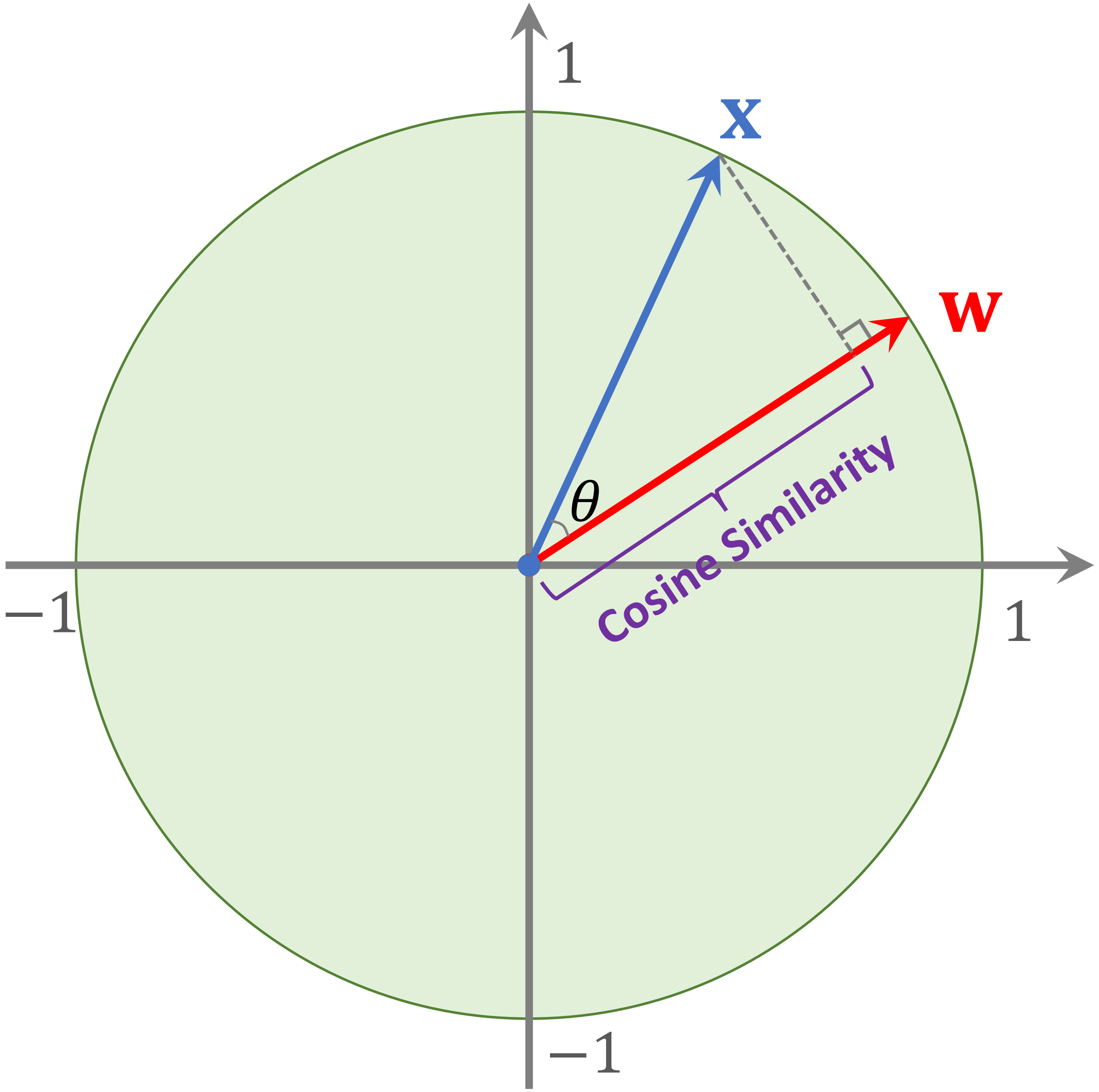
- Assume $\mathbf{x}$ and $\mathbf{w}$ are unit vectors:

$$||\mathbf{x}||_2 = 1 \text{ and } ||\mathbf{w}||_2 = 1.$$

- Cosine similarity:

$$\cos\theta = \mathbf{x}^T \mathbf{w}.$$

Cosine Similarity



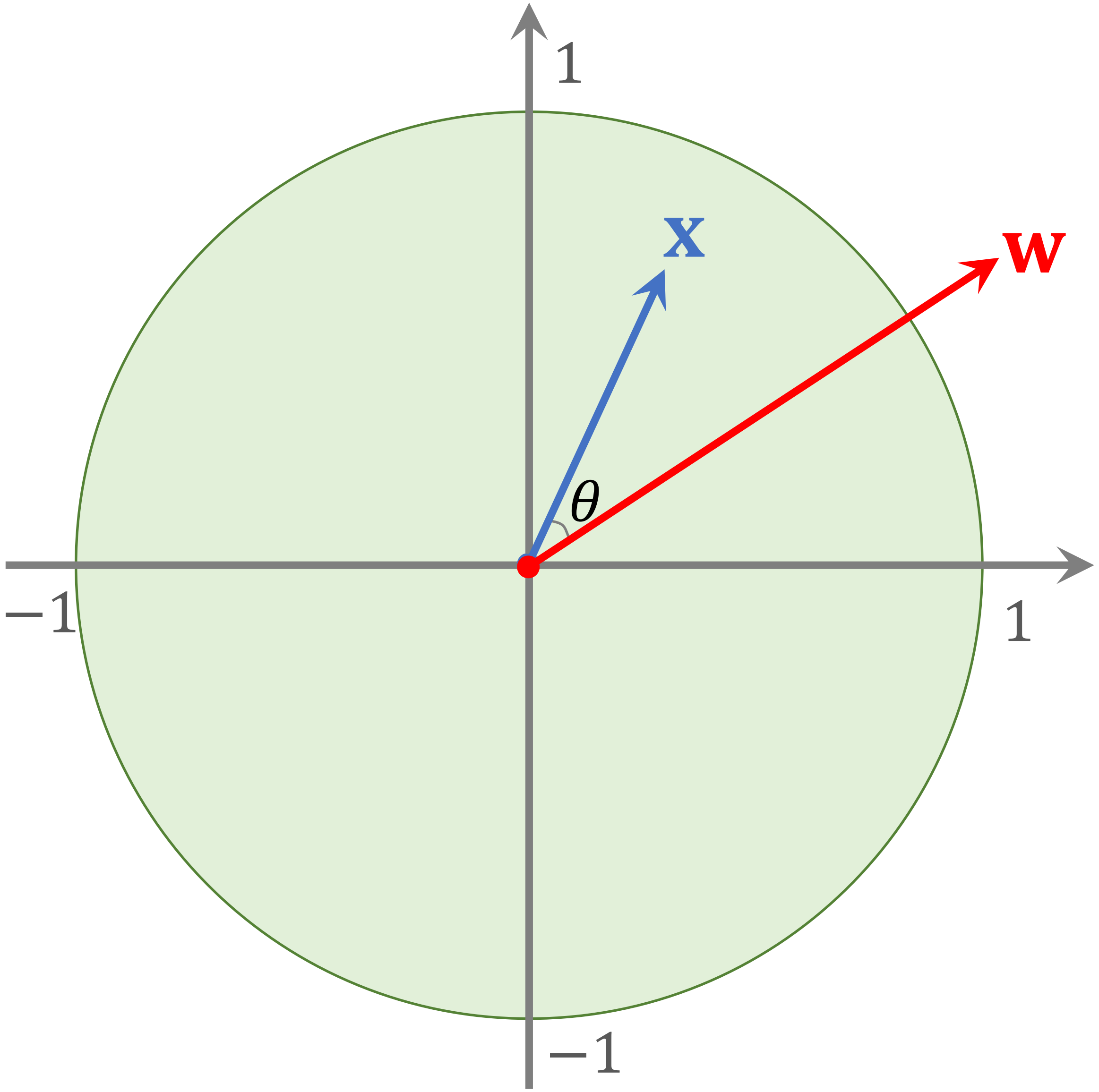
- Assume $\mathbf{x}$ and $\mathbf{w}$ are unit vectors:

$$||\mathbf{x}||_2 = 1 \text{ and } ||\mathbf{w}||_2 = 1.$$

- Cosine similarity:

$$\cos\theta = \mathbf{x}^T\mathbf{w}.$$

Cosine Similarity



- If $\mathbf{x}$ and $\mathbf{w}$ are not unit vectors, then their cosine similarity is:

$$\cos\theta = \frac{\mathbf{x}^T\mathbf{w}}{||\mathbf{x}||_2 \cdot ||\mathbf{w}||_2}.$$

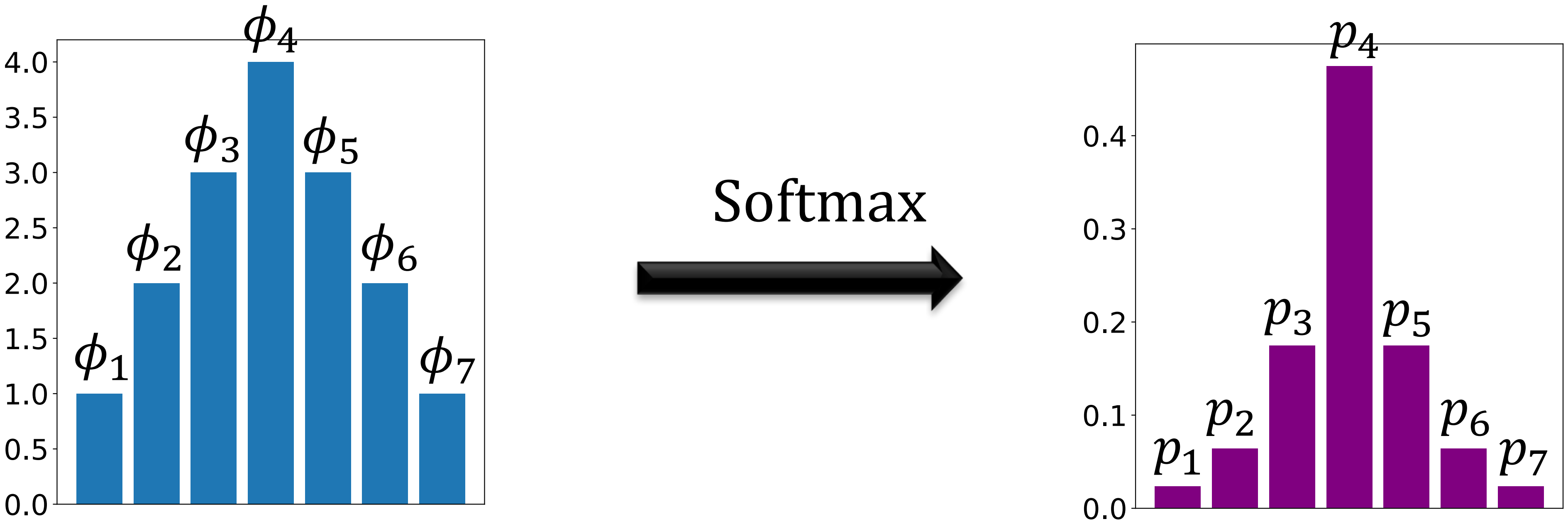
Softmax Function

- $\boldsymbol{\phi} = [\phi_1, \phi_2, \cdots, \phi_k] \in \mathbb{R}^k$.
- $\mathbf{p} = \text{normalize}\left([e^{\phi_1}, e^{\phi_2}, \cdots, e^{\phi_k}]\right) \in \mathbb{R}^k$.
- $\mathbf{p}$ is Softmax($\boldsymbol{\phi}$).

Softmax Function

- $\boldsymbol{\phi} = [\phi_1, \phi_2, \cdots, \phi_k] \in \mathbb{R}^k$.
- $\mathbf{p} = \text{normalize}\left(\left[e^{\phi_1}, e^{\phi_2}, \cdots, e^{\phi_k}\right]\right) \in \mathbb{R}^k$.
- $\mathbf{p}$ is $\text{Softmax}(\boldsymbol{\phi})$.
- Properties:
  - $p_i > 0$, for $i = 1, \cdots, k$.
  - $p_1 + p_2 + \cdots + p_k = 1$.

Softmax Function



Softmax Classifier

$$\underset{k\times 1}{\mathbf{p}} = \text{Softmax}\left[\underset{k\times d}{\mathbf{W}} \cdot \underset{d\times 1}{\mathbf{x}} + \underset{k\times 1}{\mathbf{b}}\right]$$

Here, k is number of classes, and d is number of features.

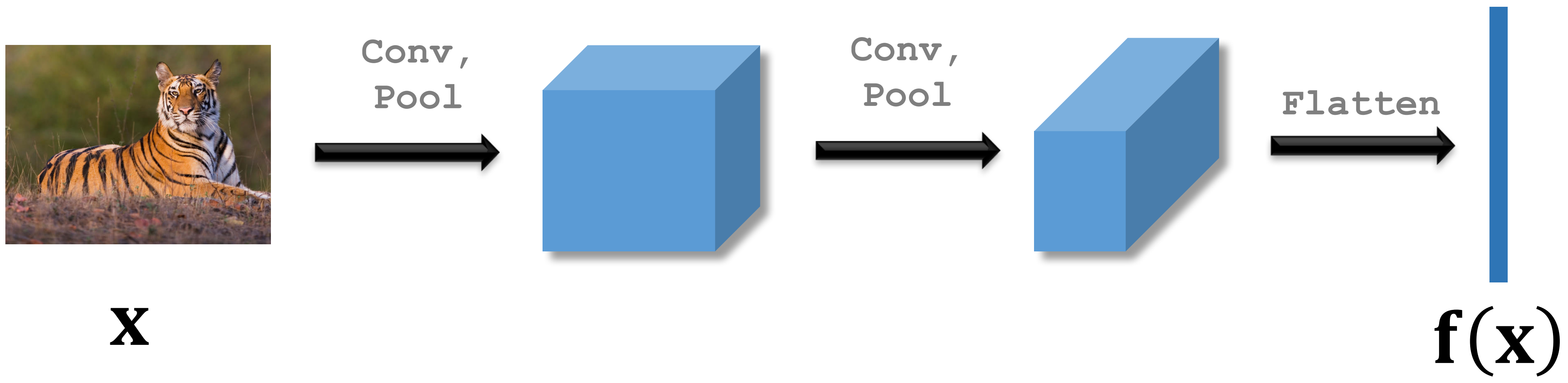
Few-Shot Prediction Using Pretrained CNN

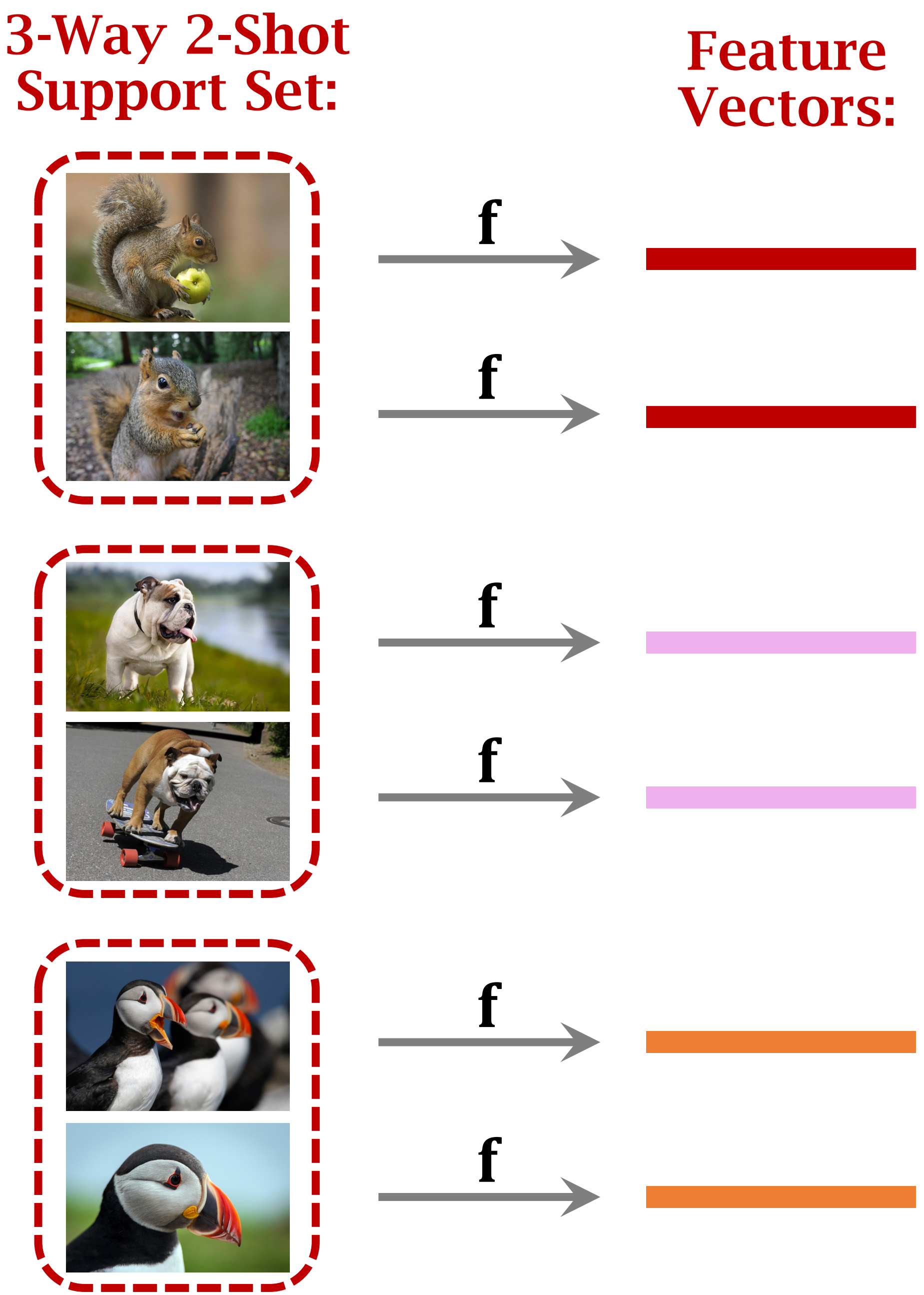
Reference:

- Dhillon, Chaudhari, Ravichandran, & Soatto. A baseline for few-shot image classification. In *ICLR*, 2020.
- Chen, Wang, Liu, Xu, & Darrell. A New Meta-Baseline for Few-Shot Learning. *arXiv*, 2020.

Pretraining

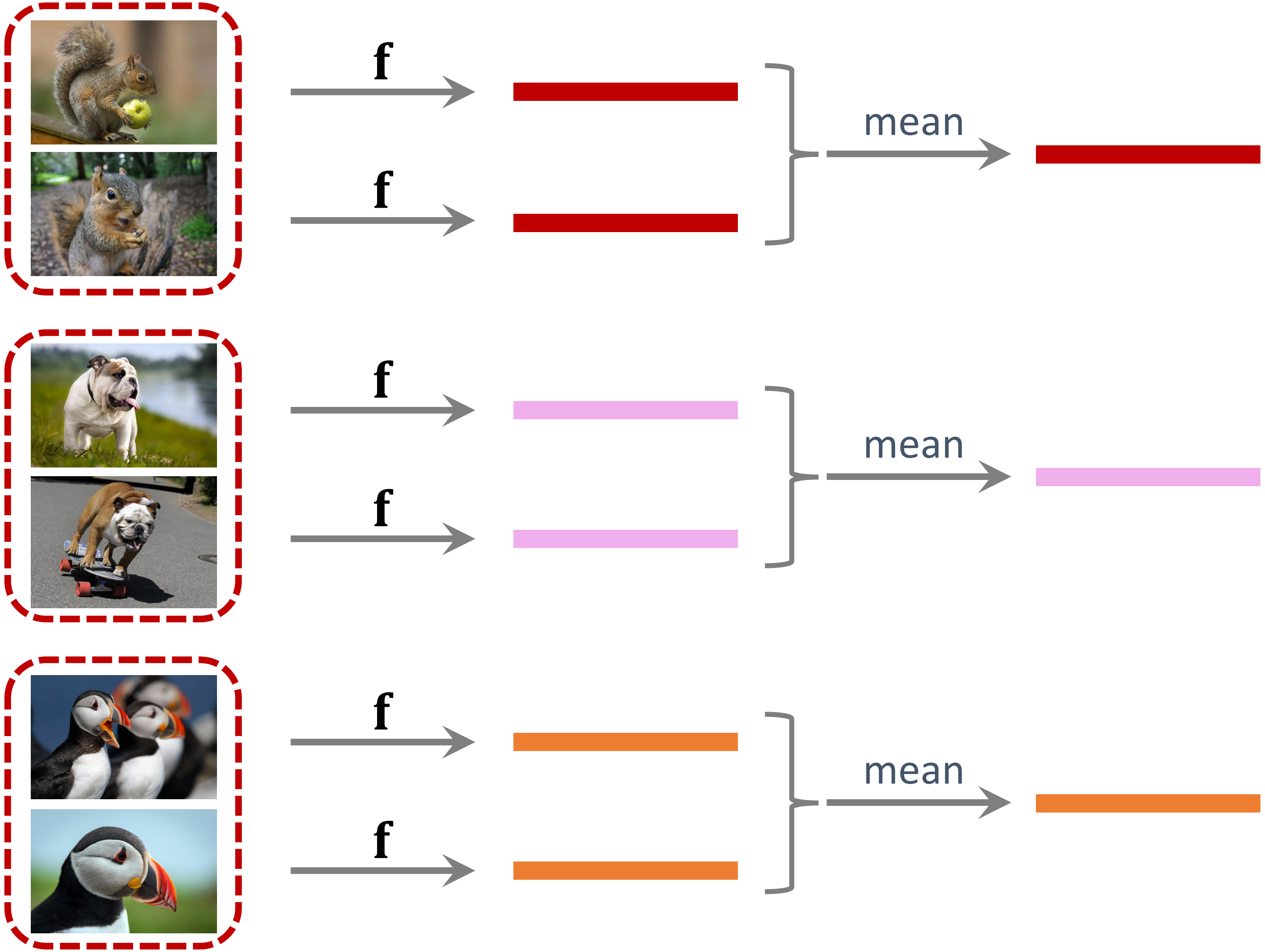
- Pretrain a CNN for feature extraction (aka embedding).
- The CNN can be pretrained using standard supervised learning or Siamese network.

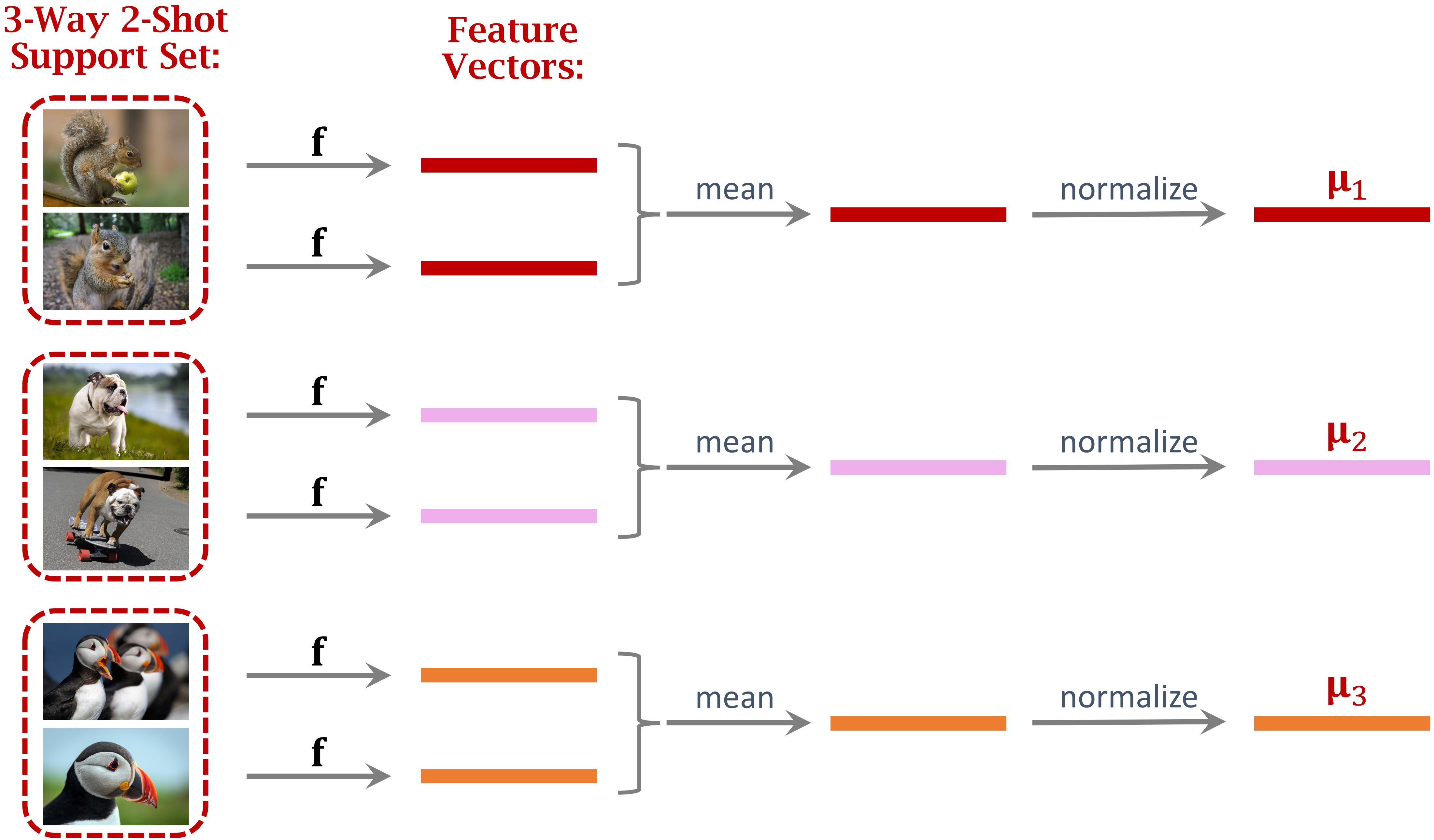



3-Way 2-Shot
Support Set:
Feature
Vectors:
f
f
f
f
f
f

3-Way 2-Shot Support Set:

Feature Vectors:

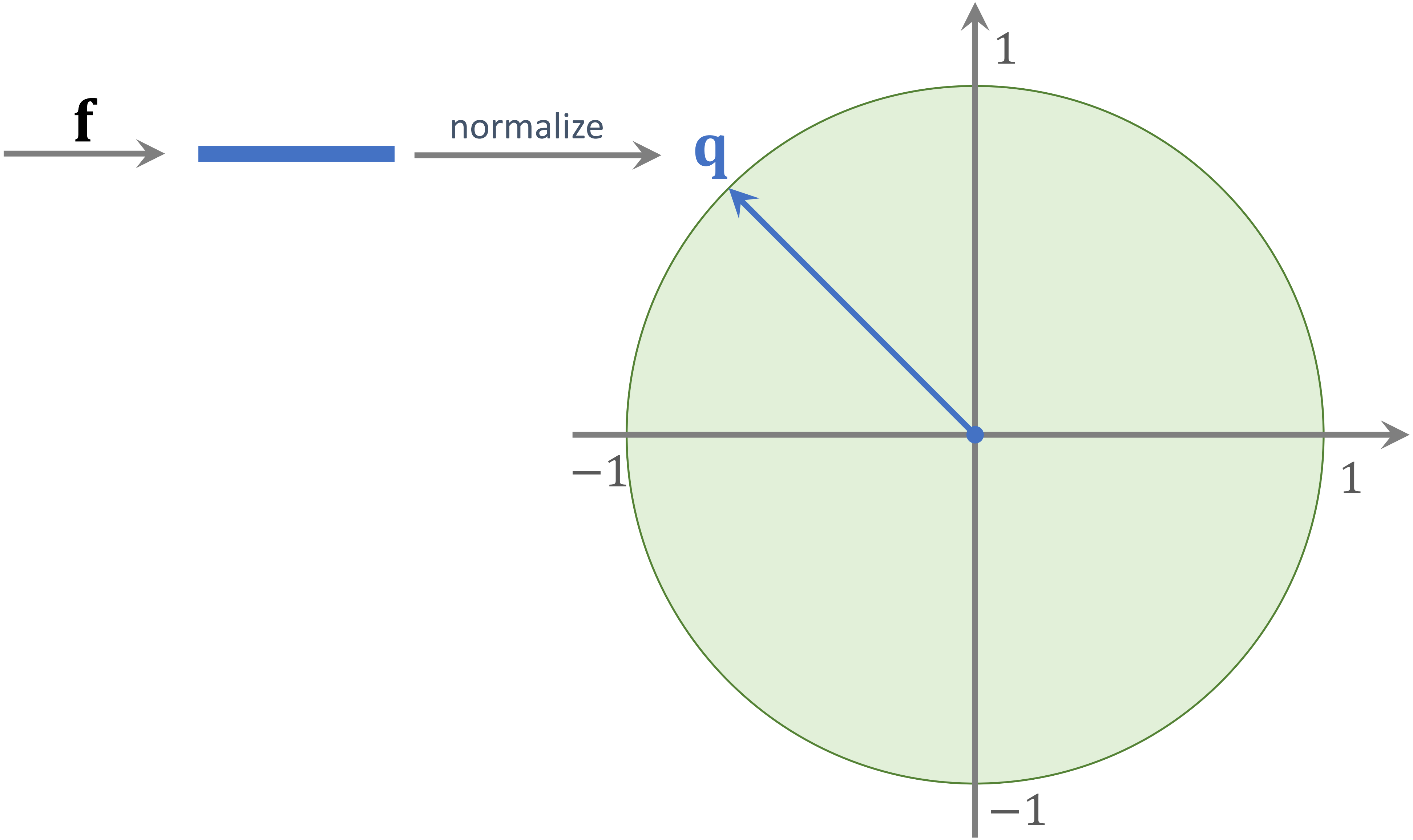




3-Way 2-Shot Support Set:
Feature Vectors:
f
mean
normalize
μ_1
μ_2
μ_3

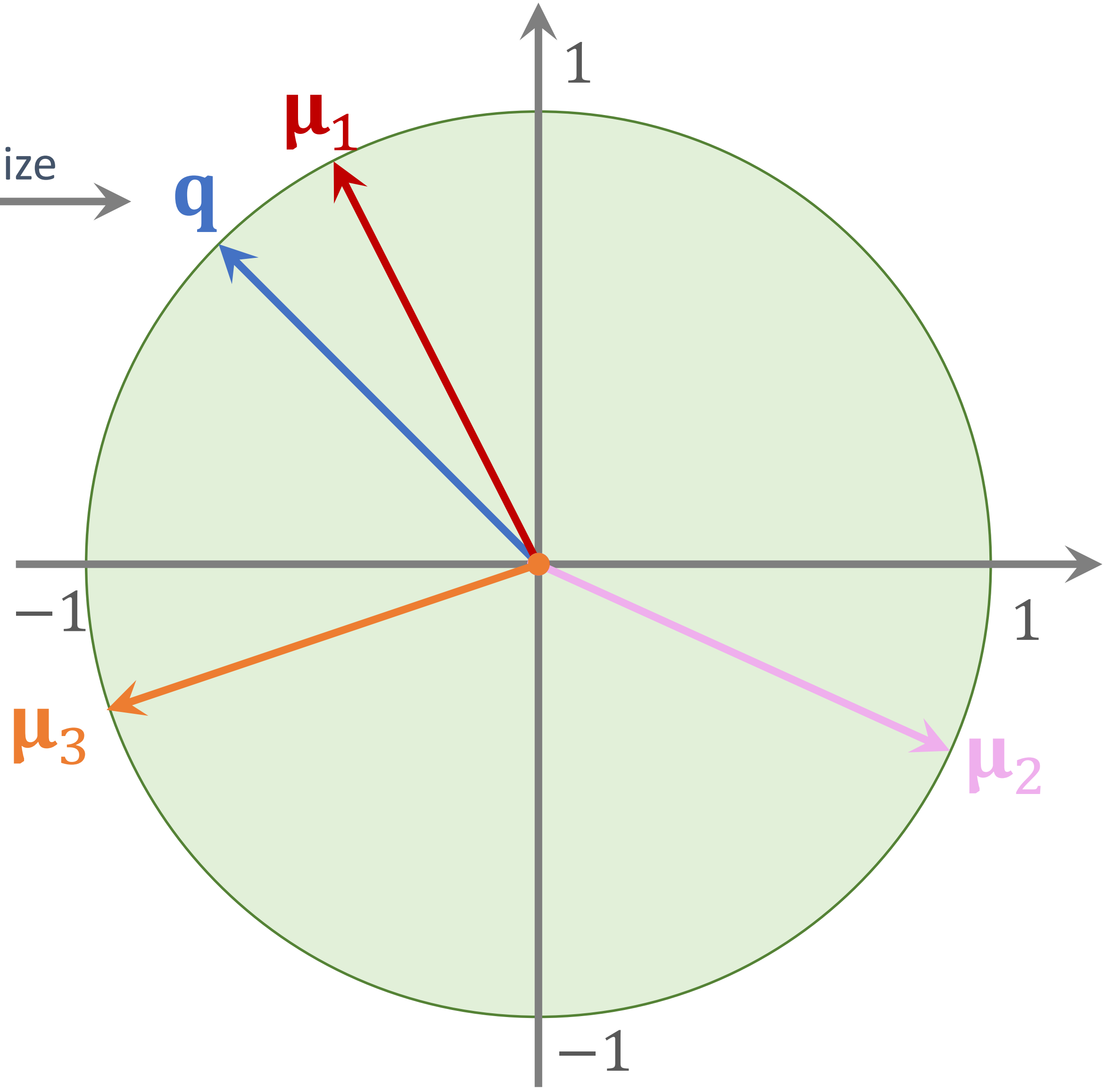
Making Few-Shot Prediction

Query



Making Few-Shot Prediction

Query



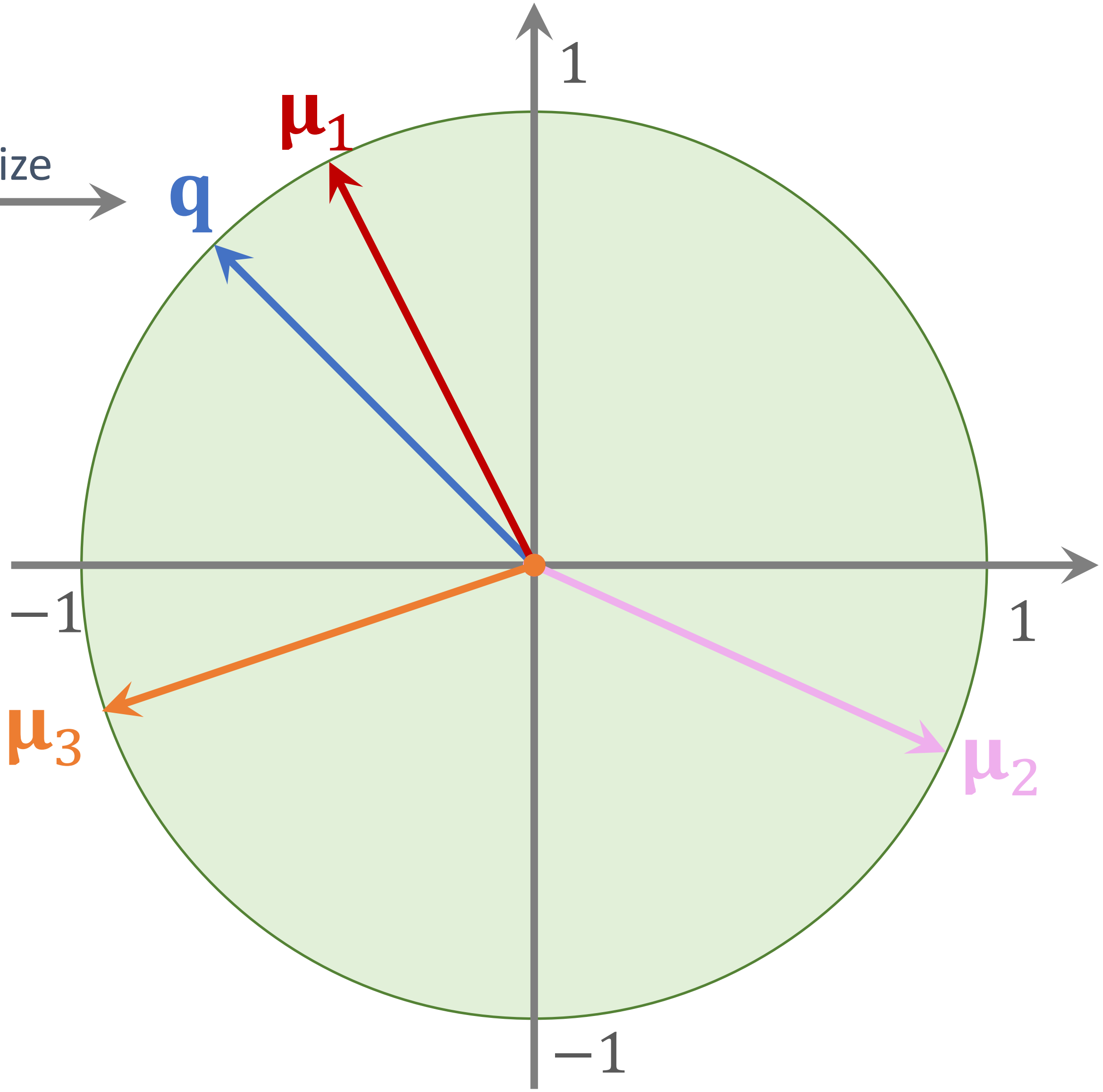
- Normalized mean feature vectors extracted from the support set:

$$M = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} q$$

$$M = \begin{bmatrix} \mu_1 q \\ \mu_2 q \\ \mu_3 q \end{bmatrix}$$

Making Few-Shot Prediction

Query



- Normalized mean feature vectors extracted from the support set:

$$\mathbf{M} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \boldsymbol{\mu}_3 \end{bmatrix}$$

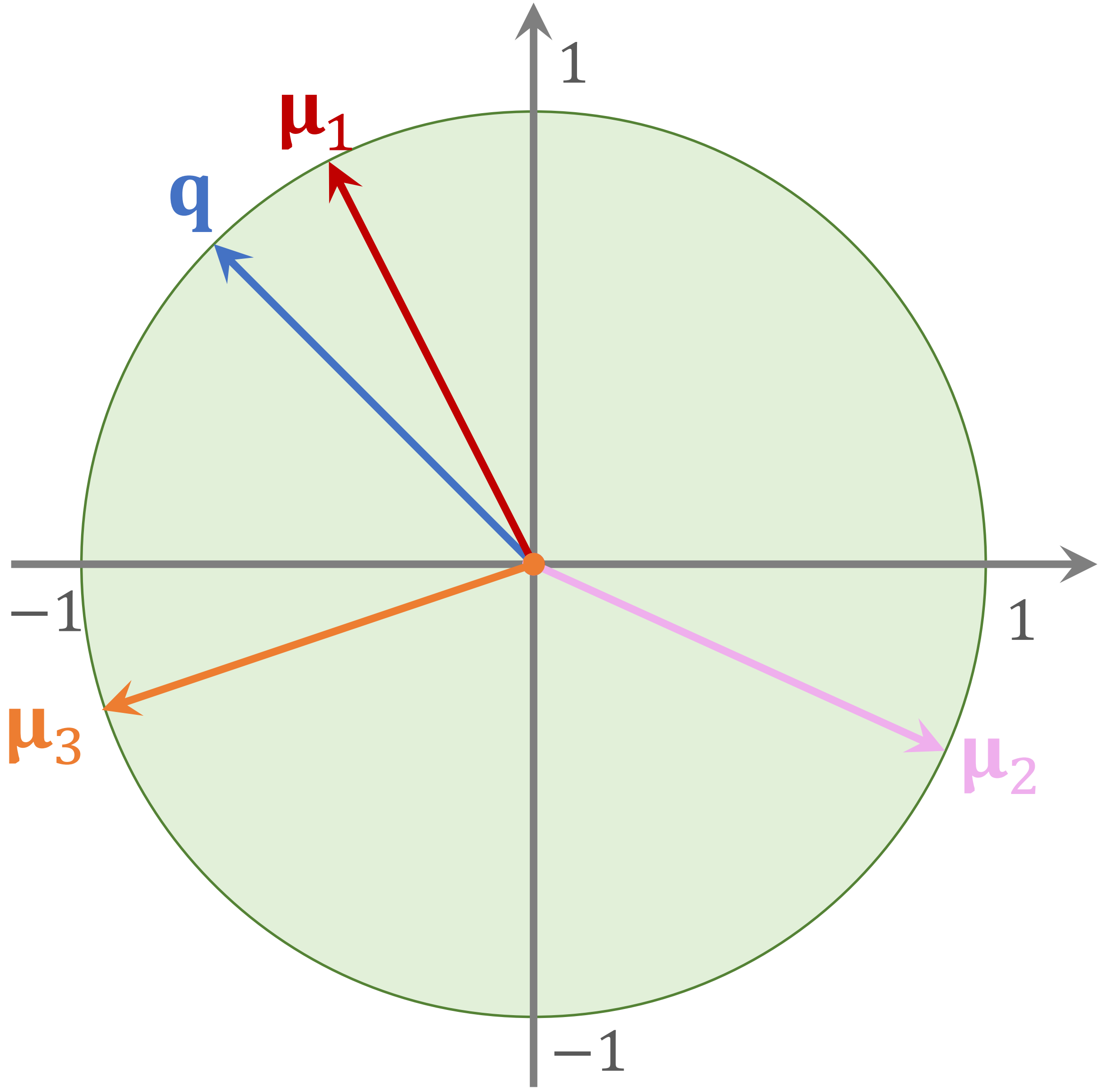
Making Few-Shot Prediction

- Make prediction:

$$\mathbf{p} = \mathrm{Softmax}(\mathbf{M}\mathbf{q})$$

$$p = \mathrm{softmax}\left(\begin{bmatrix} \mu_1^T q \\ \mu_2^T q \\ \mu_3^T q \end{bmatrix} \right)$$

= at which index p is biggest.

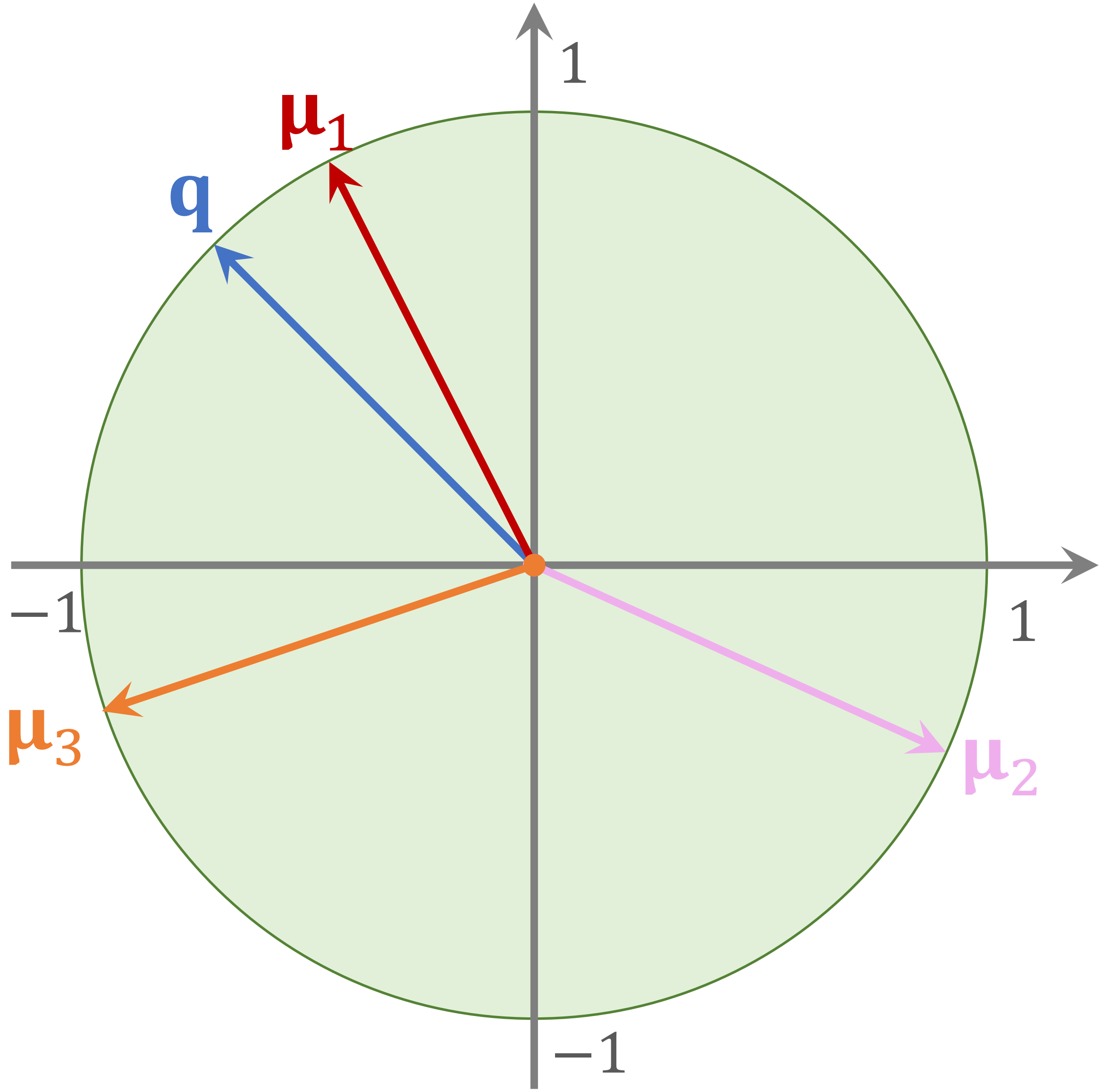


Making Few-Shot Prediction

- Make prediction:

$$\mathbf{p} = \text{Softmax}(\mathbf{M}\mathbf{q}) = \text{Softmax}\left(\begin{bmatrix} \boldsymbol{\mu}_1^T \mathbf{q} \\ \boldsymbol{\mu}_2^T \mathbf{q} \\ \boldsymbol{\mu}_3^T \mathbf{q} \end{bmatrix}\right).$$

- Which entry of $\mathbf{p}$ is the biggest?



Fine-Tuning

Reference:

- Chen, Liu, Kira, Wang, & Huang. A Closer Look at Few-shot Classification. In *ICLR*, 2019.
- Dhillon, Chaudhari, Ravichandran, & Soatto. A baseline for few-shot image classification. In *ICLR*, 2020.
- Chen, Wang, Liu, Xu, & Darrell. A New Meta-Baseline for Few-Shot Learning. *arXiv*, 2020.

Few-Shot Prediction Using Pretrained CNN

- Let $(\mathbf{x}_j, \mathbf{y}_j)$ be a labeled sample in the support set.
- $\mathbf{f}(\mathbf{x}_j)$ is the feature vector extracted by the pretrained CNN.
- $\mathbf{p}_j = \mathrm{Softmax}(\mathbf{W} \cdot \mathbf{f}(\mathbf{x}_j) + \mathbf{b})$ is the prediction.

Few-Shot Prediction Using Pretrained CNN

- Let $(\mathbf{x}_j, \mathbf{y}_j)$ be a labeled sample in the support set.
- $\mathbf{f}(\mathbf{x}_j)$ is the feature vector extracted by the pretrained CNN.
- $\mathbf{p}_j = \text{Softmax}(\mathbf{W} \cdot \mathbf{f}(\mathbf{x}_j) + \mathbf{b})$ is the prediction.

We can fix $\mathbf{W} = \mathbf{M}$ and $\mathbf{b} = \mathbf{0}$. (What we have discussed.)

$$\mathbf{M} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \boldsymbol{\mu}_3 \end{bmatrix}$$

Fine Tuning

- Let $(\mathbf{x}_j, \mathbf{y}_j)$ be a labeled sample in the support set.
- $\mathbf{f}(\mathbf{x}_j)$ is the feature vector extracted by the pretrained CNN.
- $\mathbf{p}_j = \text{Softmax}(\mathbf{W} \cdot \mathbf{f}(\mathbf{x}_j) + \mathbf{b})$ is the prediction.

We can train $\mathbf{W}$ and $\mathbf{b}$ on the support set. (Fine tuning.)

- $\min \sum_j \text{CrossEntropy}(\mathbf{y}_j, \mathbf{p}_j)$

Sum over all the samples in the support set.

Fine Tuning

- Let $(\mathbf{x}_j, \mathbf{y}_j)$ be a labeled sample in the support set.
- $\mathbf{f}(\mathbf{x}_j)$ is the feature vector extracted by the pretrained CNN.
- $\mathbf{p}_j = \text{Softmax}(\mathbf{W} \cdot \mathbf{f}(\mathbf{x}_j) + \mathbf{b})$ is the prediction.

We can train $\mathbf{W}$ and $\mathbf{b}$ on the support set. (Fine tuning.)

- $\min \sum_j \text{CrossEntropy}(\mathbf{y}_j, \mathbf{p}_j)$

Optimization variable:
- $\mathbf{W}$ and $\mathbf{b}$.
- Parameters of the CNN (optional).

Fine Tuning

- Let $(\mathbf{x}_j, \mathbf{y}_j)$ be a labeled sample in the support set.
- $\mathbf{f}(\mathbf{x}_j)$ is the feature vector extracted by the pretrained CNN.
- $\mathbf{p}_j = \text{Softmax}(\mathbf{W} \cdot \mathbf{f}(\mathbf{x}_j) + \mathbf{b})$ is the prediction.

We can train **W** and **b** on the support set. (Fine tuning.)

- $\min \sum_j \text{CrossEntropy}(\mathbf{y}_j, \mathbf{p}_j) + \text{Regularization}.$

Benefit of Fine Tuning

- Fine-tuning substantially improves the prediction accuracy [1].
  - 2% ~ 7% improvement for 5-way 1-shot.
  - 1.5% ~ 4% improvement for 5-way 5-shot.
- Similar conclusions in other papers, e.g., [2, 3].
- Comparable to the sophisticated state-of-the-art methods.

Reference:

1. Dhillon, Chaudhari, Ravichandran, & Soatto. A baseline for few-shot image classification. In *ICLR*, 2020.
2. Chen, Liu, Kira, Wang, & Huang. A Closer Look at Few-shot Classification. In *ICLR*, 2019.
3. Chen, Wang, Liu, Xu, & Darrell. A New Meta-Baseline for Few-Shot Learning. *arXiv*, 2020.

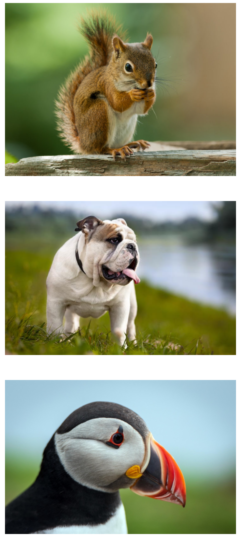
Trick 1: A Good Initialization

- Prediction made by Softmax classifier:

$$\mathbf{p} = \mathrm{Softmax}(\mathbf{W} \cdot \mathbf{f}(\mathbf{x}) + \mathbf{b}).$$

- A good initialization: $\mathbf{W} = \mathbf{M}$ and $\mathbf{b} = \mathbf{0}$ [1].

$$\mathbf{M} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \boldsymbol{\mu}_3 \end{bmatrix}$$



Reference:

1. Dhillon, Chaudhari, Ravichandran, & Soatto. A baseline for few-shot image classification. In *ICLR*, 2020.

Trick 2: Entropy Regularization

Entropy regularization is a good option [1].

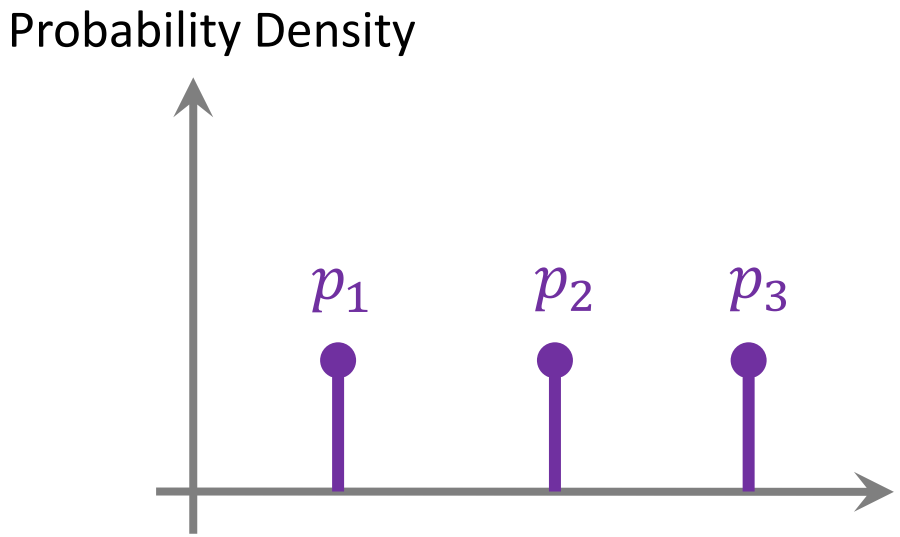
- Let $\mathbf{x}$ be a query sample.
- $\mathbf{p} = \mathrm{Softmax}(\mathbf{W} \cdot \mathbf{f}(\mathbf{x}) + \mathbf{b})$ is the prediction.
- Entropy: $\mathbb{H}(\mathbf{p}) = -\sum_i p_i \log p_i$.
- Entropy regularization: mean of $\mathbb{H}(\mathbf{p})$, for all query samples.
- Encourage the entropy regularization to be small. (Why?)

Reference:

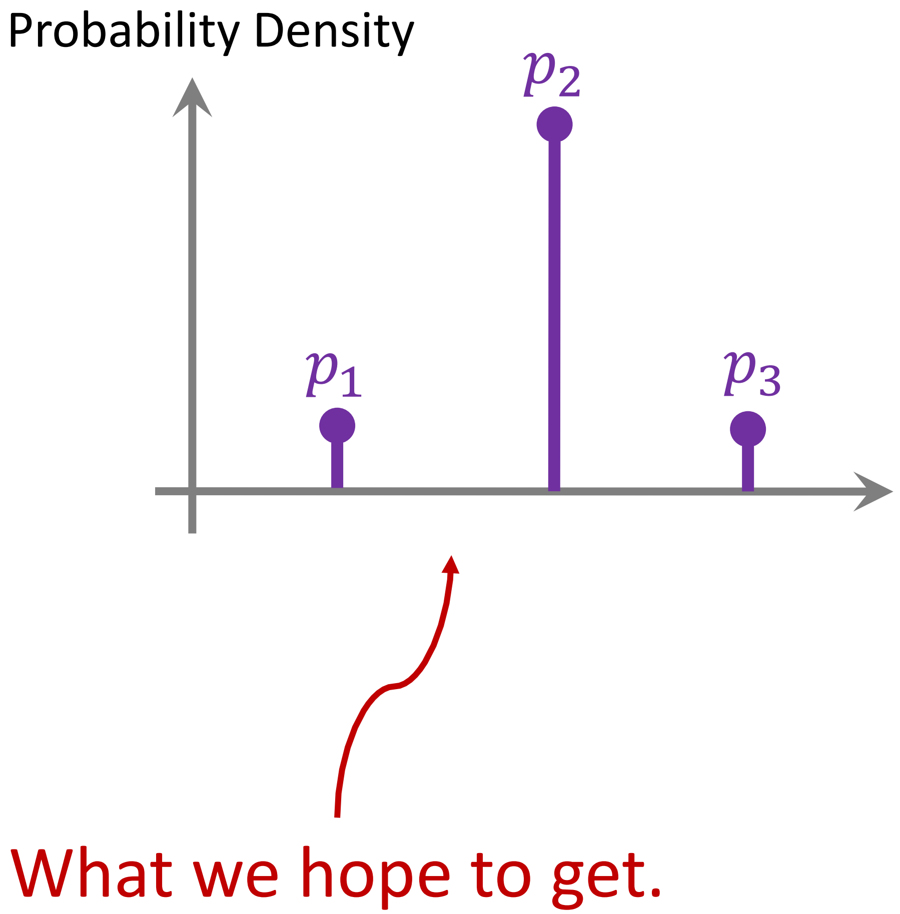
1. Dhillon, Chaudhari, Ravichandran, & Soatto. A baseline for few-shot image classification. In *ICLR*, 2020.

Trick 2: Entropy Regularization

High Entropy



Low Entropy



Trick 3: Cosine Similarity + Softmax Classifier

- Standard Softmax classifier:

$$\mathbf{p} = \mathrm{Softmax}(\mathbf{W}\,\mathbf{q} + \mathbf{b})$$

Trick 3: Cosine Similarity + Softmax Classifier

- Standard Softmax classifier:

$$\mathbf{p} = \mathrm{Softmax}(\mathbf{W}\,\mathbf{q} + \mathbf{b}) = \mathrm{Softmax}\left(\begin{bmatrix} \mathbf{w}_1^T\mathbf{q} + b_1 \\ \mathbf{w}_2^T\mathbf{q} + b_2 \\ \mathbf{w}_3^T\mathbf{q} + b_3 \end{bmatrix}\right).$$

Trick 3: Cosine Similarity + Softmax Classifier

- Standard Softmax classifier:

$$\mathbf{p} = \text{Softmax}(\mathbf{W}\,\mathbf{q} + \mathbf{b}) = \text{Softmax}\left(\begin{bmatrix} \mathbf{w}_1^T \mathbf{q} + b_1 \\ \mathbf{w}_2^T \mathbf{q} + b_2 \\ \mathbf{w}_3^T \mathbf{q} + b_3 \end{bmatrix} \right).$$

- Replacing inner product by cosine similarity:

$$\mathbf{p} = \text{Softmax}\left(\begin{bmatrix} \text{sim}(\mathbf{w}_1, \mathbf{q}) + b_1 \\ \text{sim}(\mathbf{w}_2, \mathbf{q}) + b_2 \\ \text{sim}(\mathbf{w}_3, \mathbf{q}) + b_3 \end{bmatrix} \right),$$

where $\text{sim}(\mathbf{w}, \mathbf{q}) = \frac{\mathbf{w}^T \mathbf{q}}{||\mathbf{w}||_2 \cdot ||\mathbf{q}||_2}$ is the cosine similarity.

Trick 3: Cosine Similarity + Softmax Classifier

- Standard Softmax classifier:

$$\mathbf{p} = \text{Softmax}(\mathbf{W}\,\mathbf{q} + \mathbf{b}) = \text{Softmax}\left(\begin{bmatrix} \mathbf{w}_1^T\mathbf{q} + b_1 \\ \mathbf{w}_2^T\mathbf{q} + b_2 \\ \mathbf{w}_3^T\mathbf{q} + b_3 \end{bmatrix}\right).$$

- Replacing inner product by cosine similarity:

$$\mathbf{p} = \text{Softmax}\left(\begin{bmatrix} \text{sim}(\mathbf{w}_1, \mathbf{q}) + b_1 \\ \text{sim}(\mathbf{w}_2, \mathbf{q}) + b_2 \\ \text{sim}(\mathbf{w}_3, \mathbf{q}) + b_3 \end{bmatrix}\right),$$

where $\text{sim}(\mathbf{w}, \mathbf{q}) = \frac{\mathbf{w}^T\mathbf{q}}{||\mathbf{w}||_2 \cdot ||\mathbf{q}||_2}$ is the cosine similarity.

Summary

Few-Shot Prediction by Pretrained CNN

Step 1: Pretraining

- Pretrain a CNN on large-scale training data.
- Use the CNN for feature extraction.

Step 2: Few-Shot Prediction

- Map images in the support set to feature vectors.
- Obtain the mean feature vector of each class: $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k$.
- Compare the feature of query with $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k$.

Few-Shot Prediction by Pretrained CNN

Step 1: Pretraining

- Pretrain a CNN on large-scale training data.
- Use the CNN for feature extraction.

Step 2: Few-Shot Prediction

- Map images in the support set to feature vectors.
- Obtain the mean feature vector of each class: $\mathbf{\mu}_1, \cdots, \mathbf{\mu}_k$.
- Compare the feature of query with $\mathbf{\mu}_1, \cdots, \mathbf{\mu}_k$.

Few-Shot Prediction by Pretrained CNN

Step 1: Pretraining

- Pretrain a CNN on large-scale training data.
- Use the CNN for feature extraction.

Step 2: Few-Shot Prediction

- Map images in the support set to feature vectors.
- Obtain the mean feature vector of each class: $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k$.
- Compare the feature of query with $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k$.

Few-Shot Prediction by Pretrained CNN

Step 1: Pretraining

Step 2: Few-Shot Prediction

Fine Tuning

Step 1: Pretraining

Step 2: Fine Tuning

- Training a classifier on the support set.
- Tricks:
  1. Using $\mathbf{M}$ to initialize $\mathbf{W}$.
  2. Entropy regularization.
  3. Cosine similarity + Softmax classifier.

Step 3: Few-Shot Prediction

Thank you!