Image Classification

Computer Vision – Lecture 06

Further Reading

- Slides from [A Zisserman and A Vedaldi]
- Pattern Recognition and Machine Learning, C Bishop
- Deep Learning, Goodfellow, Bengio, Courville

So far

- We know a/one way to compare images:
  - Compute a BoW descriptor for each image.
  - This allows finding similar images: compute descriptor similarities
- Does this tell us if the image contains a cat?
- How do we know if the image contains a cat?

Image Embeddings

- Formally, we define a feature extractor ϕ: $\mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^{d}$ for images.
- ϕ maps images to d-dimensional descriptor vectors.
- A good ϕ maps similar images close-by in the feature space, while different image have large distances.
- BoW is an image embedding.
- Allows retrieval and many other applications.

Supervised Learning Summary

Dataset $D = \{(x_i, y_i) | 1 \leq i \leq N\}$

Inputs x_i

Outputs y_i

Training/Validation/Testing $D = D_T \cup D_V \cup D_*$

Learn $f(x) = y$ by minimizing $\sum_{(x_i,y_i) \in D_T} \mathcal{L}(f(x_i), y_i)$

Hoping to generalise: $\sum_{(x_i,y_i) \in D_*} \mathcal{L}(f(x_i), y_i)$

Supervised Learning - Data

CIFAR-10 dataset (Alex Krizhevsky, 2009)

- 60000 32x32 colour images
- 10 classes
- 6000 images per class
- 50000 training images
- 10000 test images

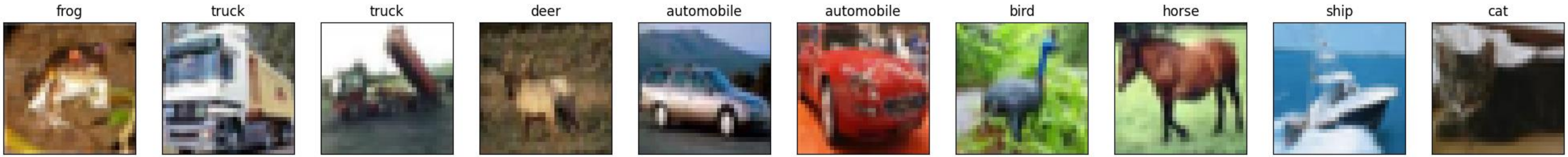


Image Embeddings

$\phi($  $)$

$\phi($  $)$

$\phi($  $)$

$\phi($  $)$

$\phi($  $)$

$\phi($  $)$

$\phi($  $)$

$\phi($  $)$

$\phi($  $)$

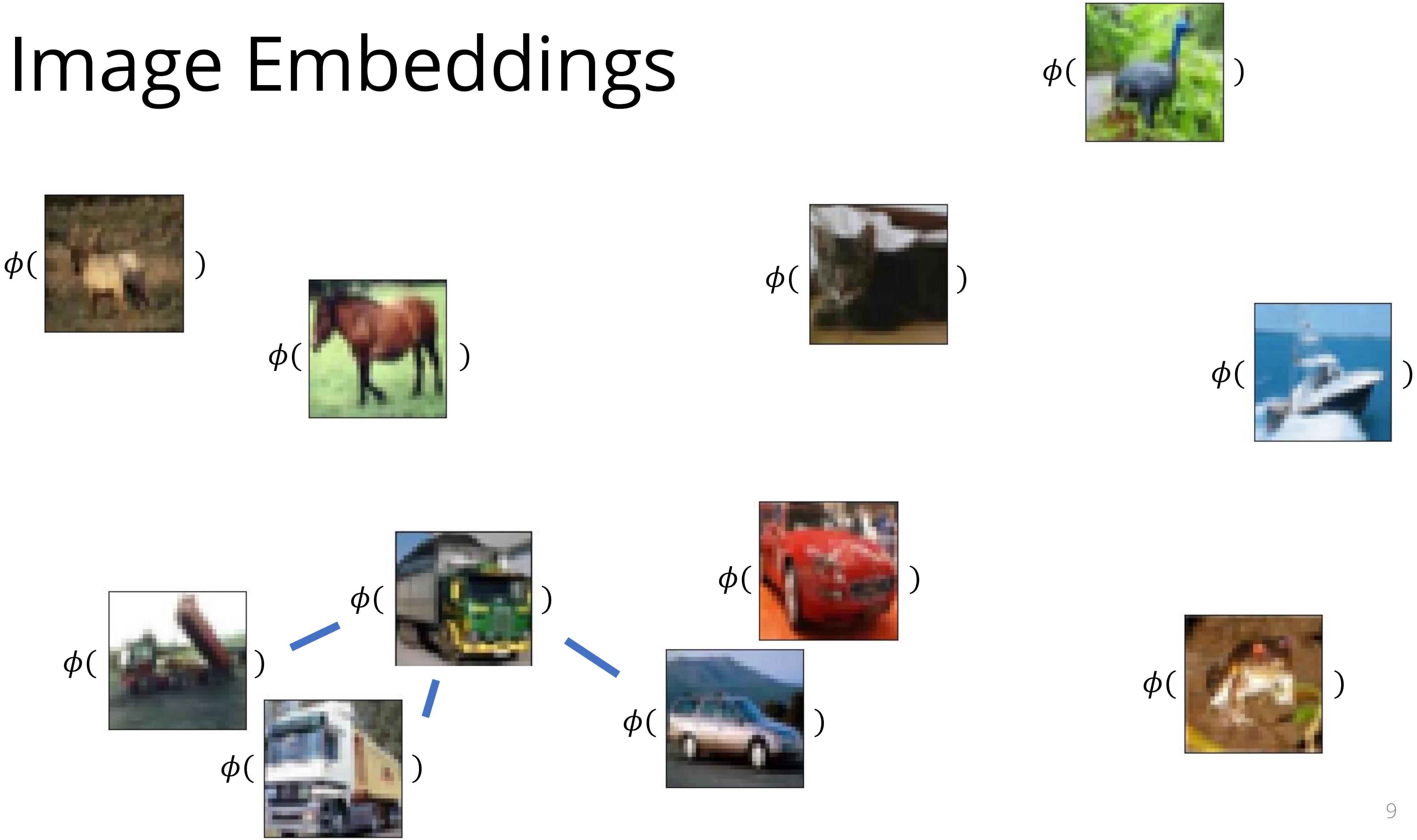
$\phi($  $)$

Nearest Neighbour Classification

- Embed a new sample with ϕ.
- Look up nearest neighbours in the embedding space.
- Predicted class is the majority vote of the neighbourhood.
- Can also return class distribution.

$\phi($  $)$

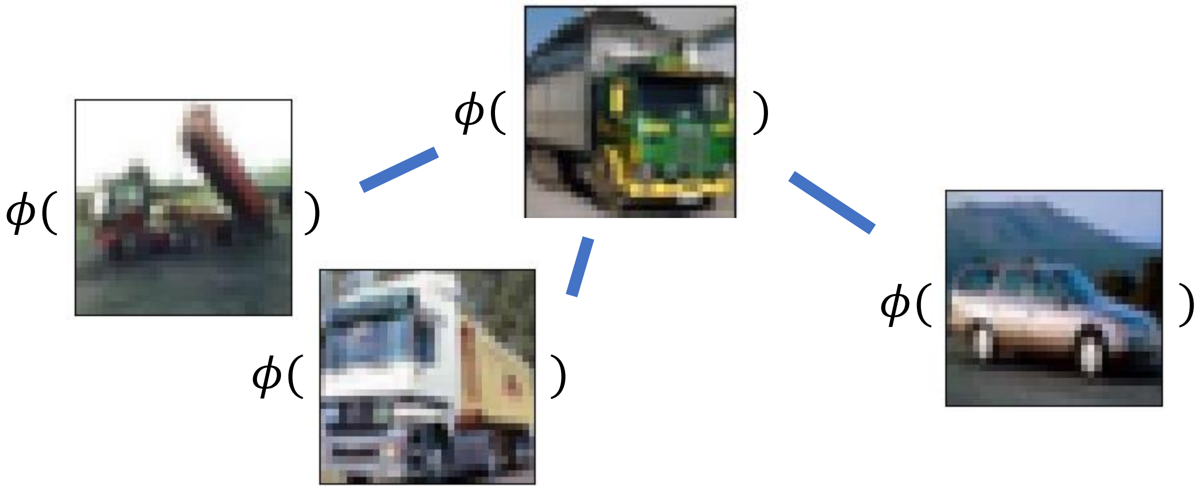
Image Embeddings



Nearest Neighbour Classification

3-NN Classification:

- $p(I, \text{truck}) = \frac{2}{3}$
- $p(I, \text{car}) = \frac{1}{3}$
- $p(I, \text{other classes}) = 0$



Nearest Neighbour Classification

Algorithm

Training:

- Precompute the feature embedding of all training samples.
- Optional: use a fast NN-lookup data structure (e.g. kd-trees).

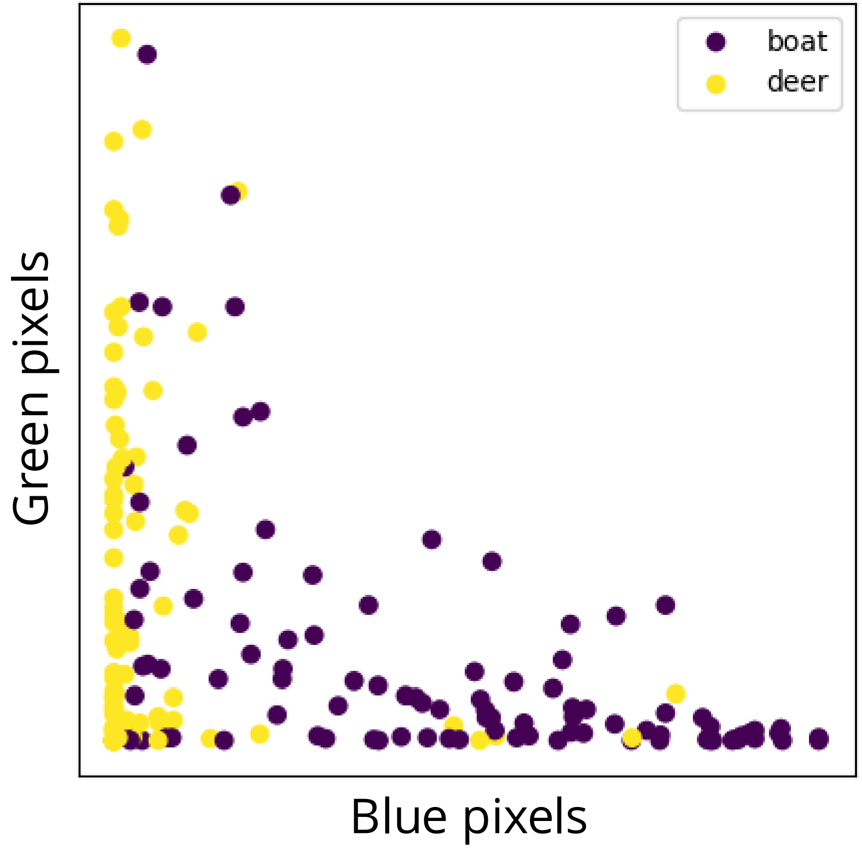
Testing:

- Compute the embedding for the new image.
- Look up k-NN and compute class histogram.

Embedding Function Example

- We count the number of blue and green pixels.
- ϕ: $\mathbb{R}^{H \times W \times 3} \to \mathbb{R}^2$ is easy to visualise.
- We classify boats vs deer.

- How do we measure the quality of a classifier?



Accuracy

We compute the number of samples the classifier f predicts correctly.

$$\mathrm{Acc}(f) = \frac{1}{|D_*|} \sum_{x,y \in D_*} [y = f(x)]$$

When the classifier predicts probabilities, we can also compute the expected accuracy:

$$\mathrm{EAcc}(f) = \frac{1}{|D_*|} \sum_{x,y \in D_*} f_y(x)$$

Where $f_y(x)$ is the predicted probability for class y.

Precision and Recall

- Accuracy can be misleading when the label distribution is skewed.
- In a dataset where 90% of samples are of class 0, you can obtain 90% accuracy by always predicting 0.
- Precision: TP/(TP+FP)
- Recall: TP/(TP+FN)

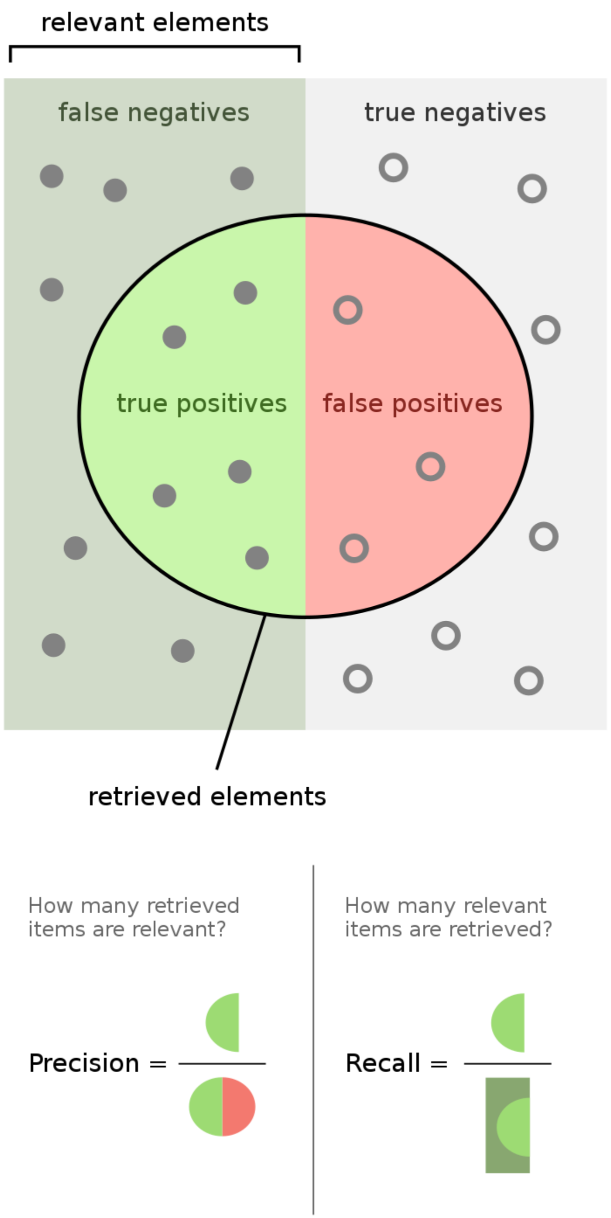


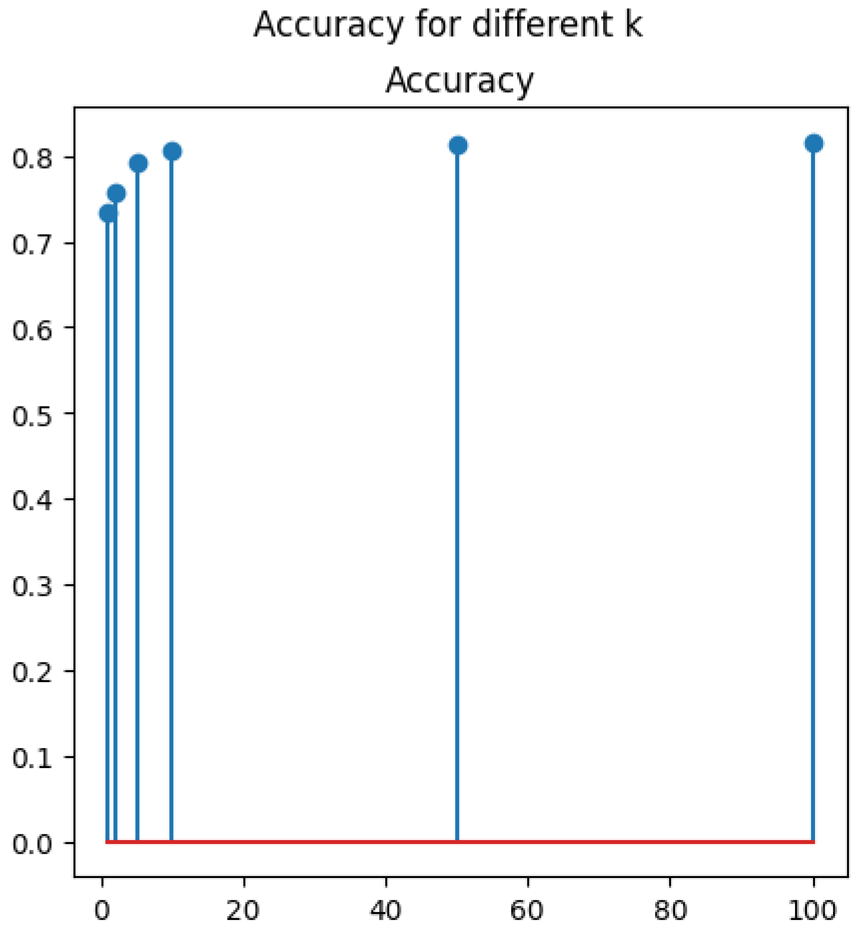
Image source

k-NN Classification

As k increases:

- Classification boundary becomes smoother.
- Might improve or worsen performance.

Choose optimal k on the validation set!



Boats vs. Deer

k-NN Summary

- k -NN is simple but effective.
- Applies to multi-class classification.
- Decision surfaces are non-linear.
- Quality of predictions automatically improves with more "training" data.
- Only a single parameter, k; easily tuned by cross-validation.
- Often used as a baseline classifier.

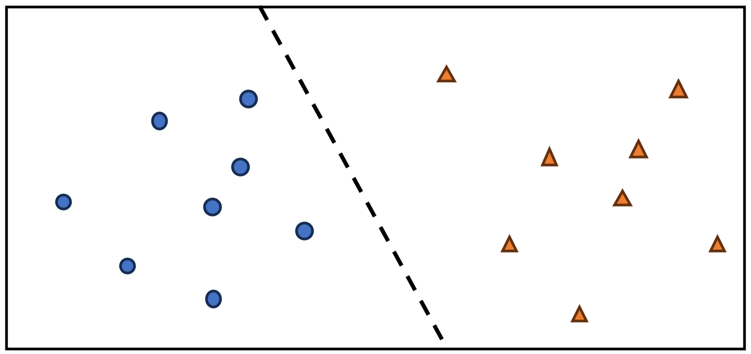
Image Classification in 2 Steps

1. Compute image embeddings.
2. Learn a classifier on the training set.

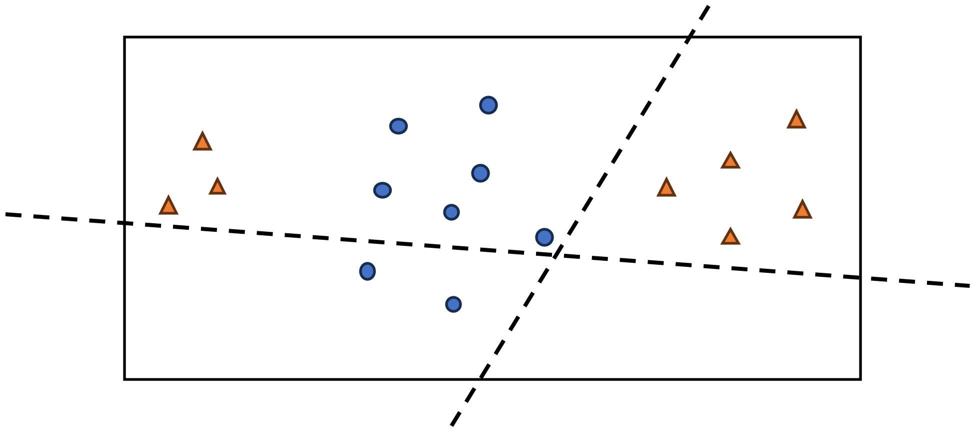
Two directions for improvements:

- Find a better embedding function.
- Find a better classifier.

Linear Separability



Linearly separable

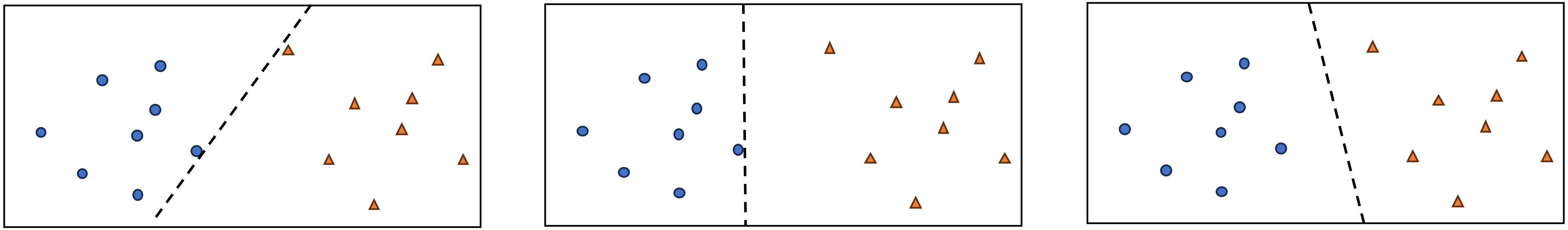


Not linearly separable

Decision Boundaries

- Even if our data is linearly separable, we might have many choices to place the decision boundary.

Intuitions



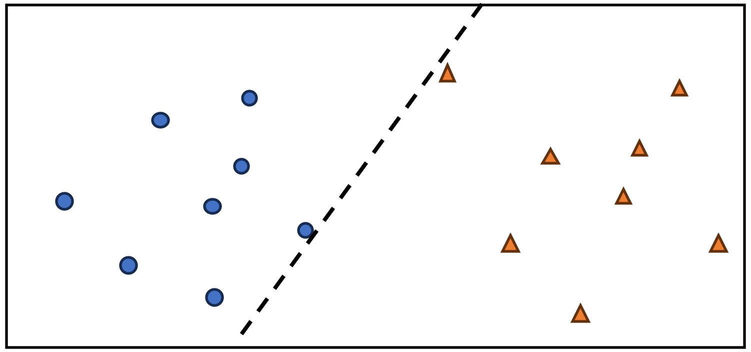
good boundary – but why?

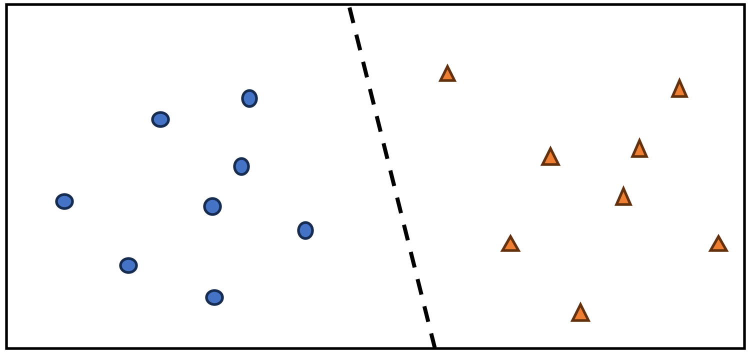
Maximum Margin

- If we assume some noise, tight boundaries can easily lead to miss-classifications.

- Both classifiers on the right have 100% accuracy.

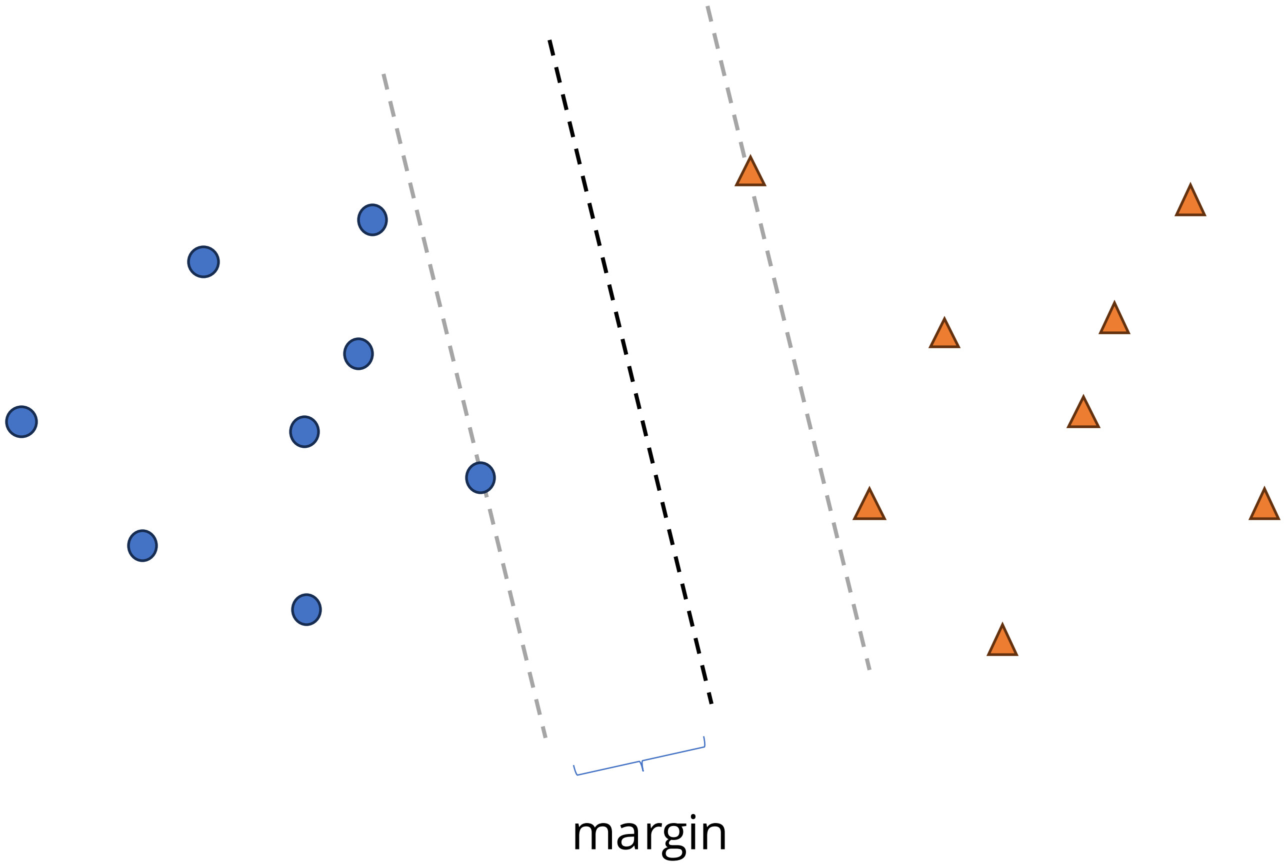
- Maximum margin idea: place the decision boundary as far away from the samples as possible.





Maximum Margin

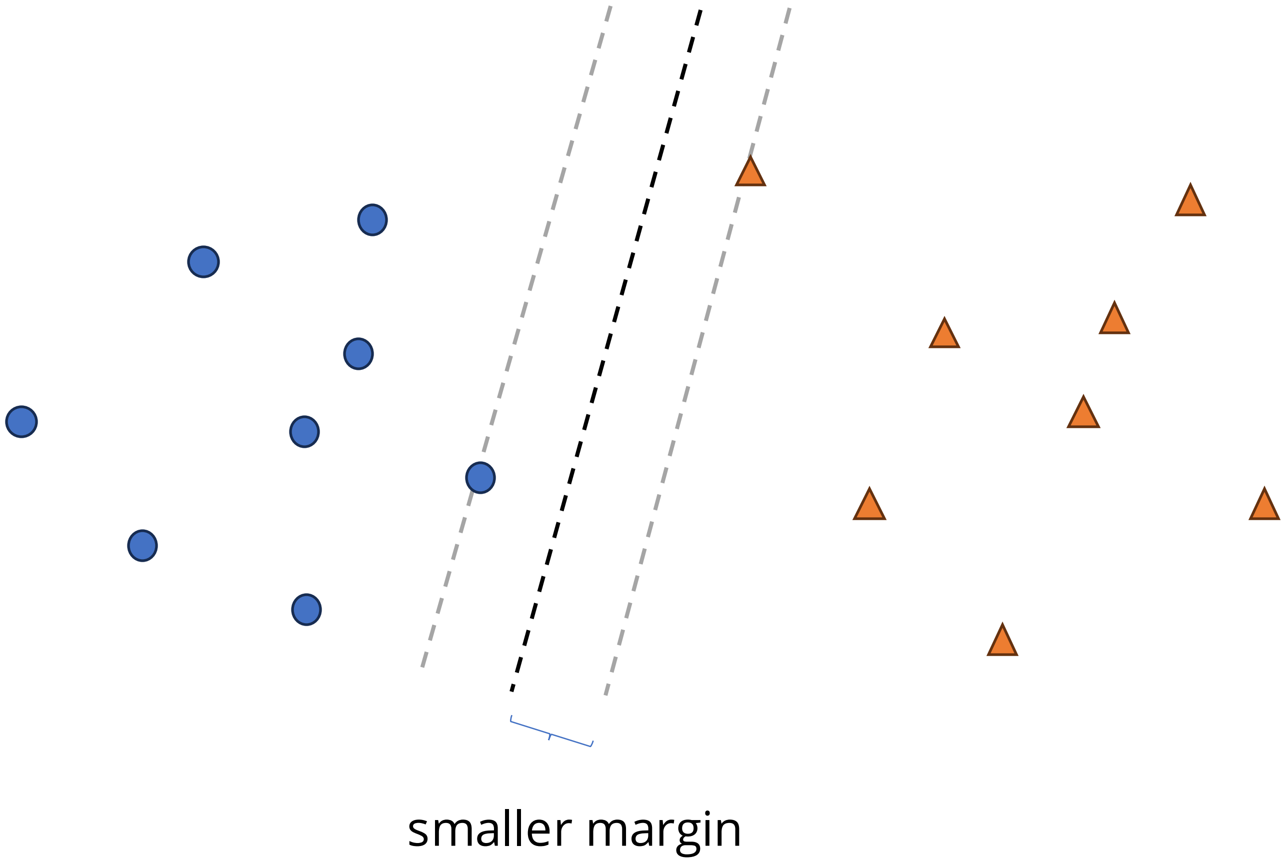
- Data points become *support vectors* for the decision boundary margins.
- We want to maximise the margin.



Maximum Margin

- Data points become *support vectors* for the decision boundary margins.

- We want to maximise the margin.



Linear Support Vector Machine

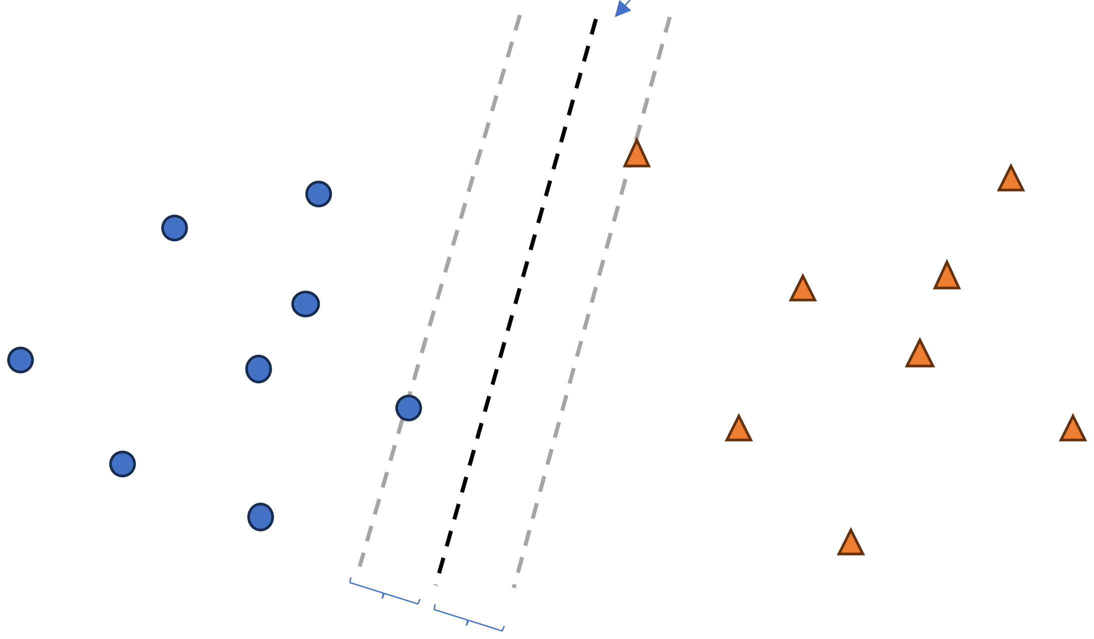
- Binary classification dataset:

$$(x_i, y_i), \qquad 1 \le i \le n, \qquad y_i \in \{-1, 1\}$$

Two objectives:

- Classification accuracy
- Maximising margins

- Linear classifier: $f(x) = w^T x + b$



Linear Support Vector Machine

Classification criterion:

$$\sum_i y_i f(x_i) = \sum_i y_i\left(x_i^T w + b\right) \geq 1$$

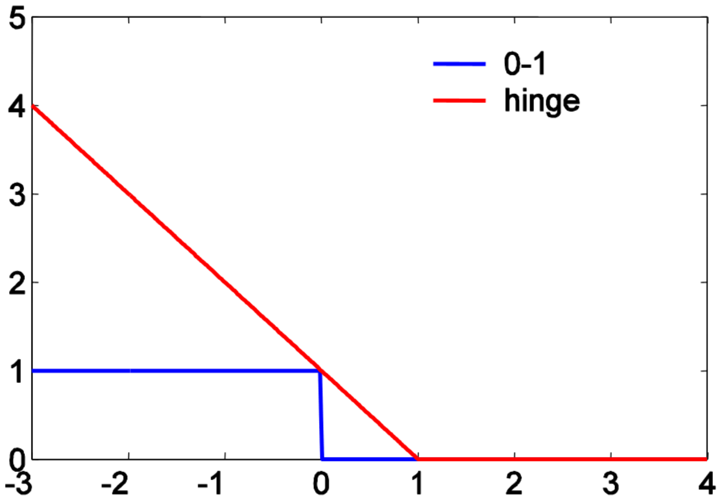
This means that y_i and $f(x_i)$ should have the same sign (=class) and $|f(x_i)| \geq 1$ pushes points beyond the margin.

Margin maximisation:

$$\min\|w\|^2$$

Quadratic optimization problem with linear constraints. In general: there is a unique solution.

Training SVMs



Constraint: $\sum_i y_i\left(x_i^T w + b\right) \geq 1$

Maximum margin: $\min||w||^2$

Training loss: $\frac{1}{\mathrm{n}}\sum_i \underbrace{\max(0,1 - y_i\left(x_i^T w + b\right))}_{\text{Hinge-loss}} + \lambda||w||^2$

Hinge loss: gradient of -1 until constraint is fulfilled.

SVM Summary

- SVMs are good linear classifiers: maximum margin decision boundaries.

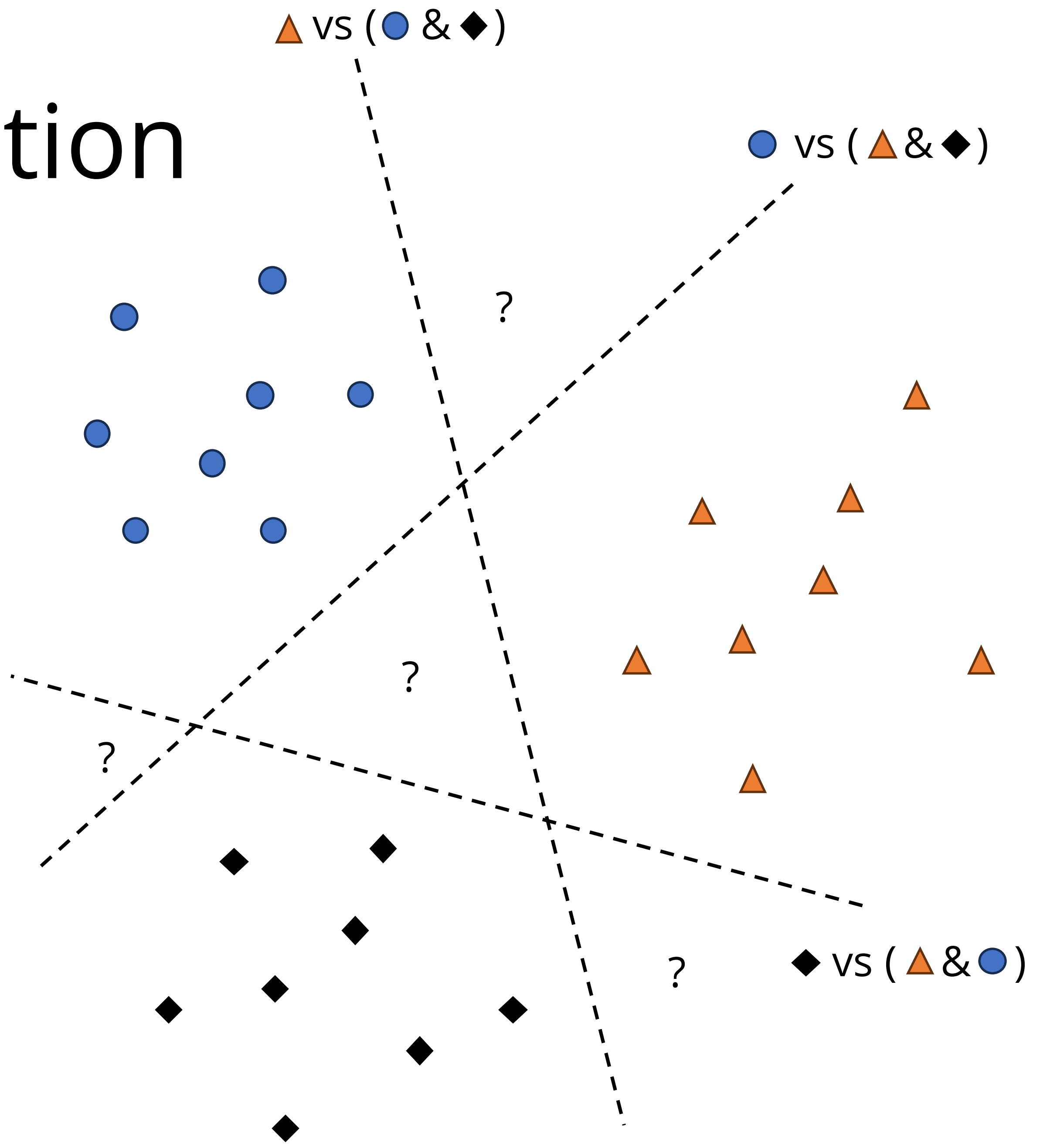
- Performance depends on the linear separability of the samples, thus on the image embedding function!

- Kernel SVM: non-linear decision boundaries.
- Intuition: learn a non-linear mapping to a space where classes are linearly separable together with the SVM

Multi-Class Classification

- Setting: $K > 2$ classes.
- Idea: train K classifiers $f_k(x)$, each is a binary 1-vs-all classifier.
- Classification: choose the class with the highest score

$$\underset{k}{\operatorname{argmax}}\, f_k(x)$$

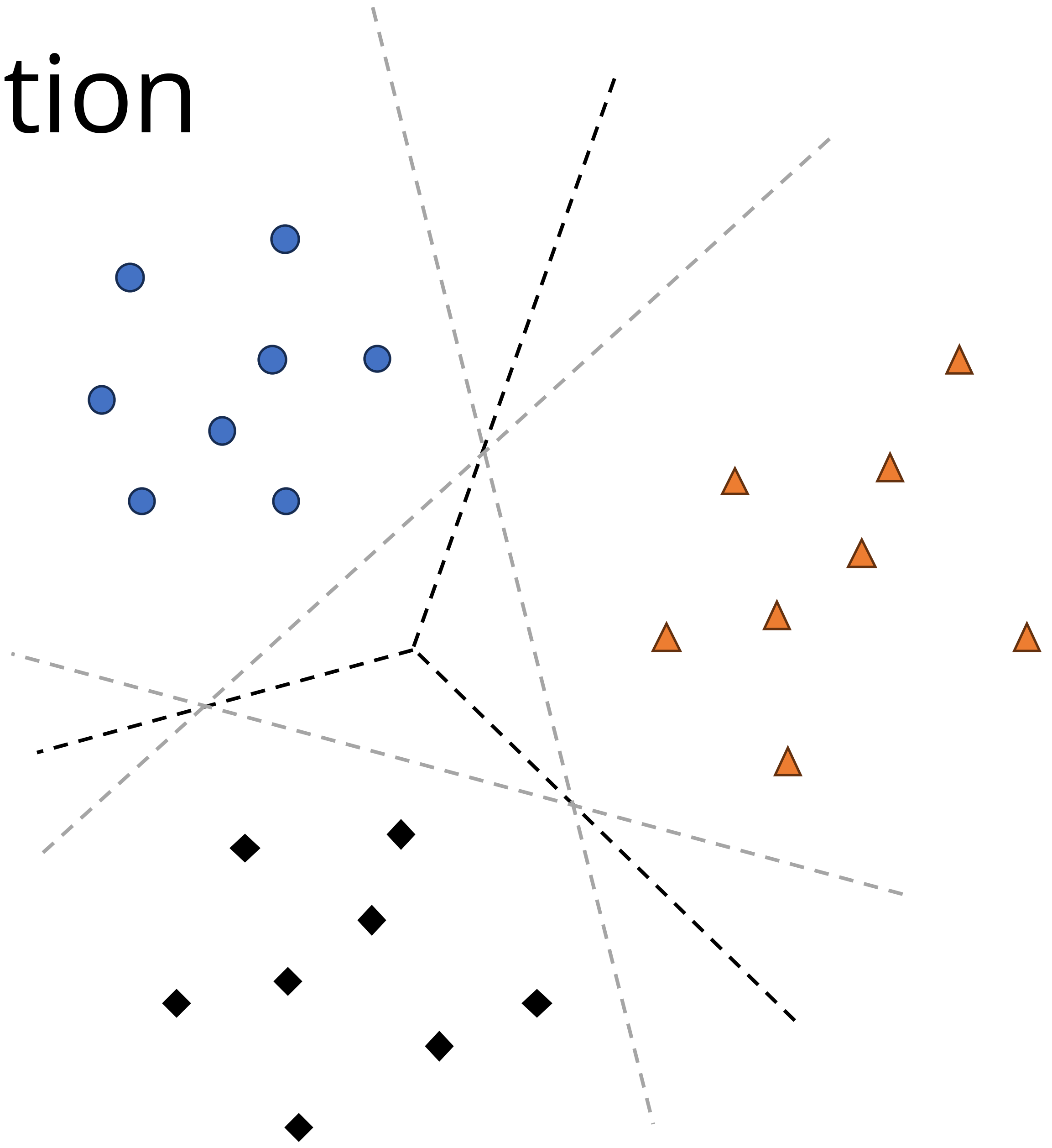


Multi-Class Classification

- Setting: $K > 2$ classes.
- Idea: train K classifiers $f_k(x)$, each is a binary 1-vs-all classifier.

- Classification: choose the class with the highest score

$$\operatorname*{argmax}_k f_k(x)$$



Multi-Class Classification

Linear classifiers

$$f_k(x) = w_k^T x + b$$

Vector form:

$$f(x) = \begin{pmatrix} w_1^T \\ \vdots \\ w_K^T \end{pmatrix} x + \begin{pmatrix} b_1 \\ \vdots \\ b_K \end{pmatrix} = Wx + B = \begin{pmatrix} f_1(x) \\ \vdots \\ f_K(x) \end{pmatrix} = \hat{Y}$$

How to we turn the class scores $\hat{Y}$ into a single class prediction?

The soft-max Function

$\operatorname*{argmax}_k \hat{Y}_k$ is not differentiable.

Idea: convert $\hat{Y}$ into a probability distribution. All elements in (0, 1) and sum to 1.

$$\mathrm{softmax}_k(\hat{Y}) = \frac{\exp \hat{Y}_k}{\sum_j \exp \hat{Y}_j}$$

The soft-max Function

$$\mathrm{softmax}_k(\hat{Y}) = \frac{\exp \hat{Y}_k}{\sum_j \exp \hat{Y}_j}$$

- Result sums to 1.
- All values between 0 and 1.
- Works for any input in $\mathbb{R}^K$ (positive and negative).
- If one $\hat{Y}_i \gg \hat{Y}_j$ than all others, softmax “selects” this element: $\mathrm{softmax}_i(\hat{Y}) \approx 1$ and $\mathrm{softmax}_j(\hat{Y}) \approx 0$.

Temperature

Often a temperature parameter τ is added to the softmax function

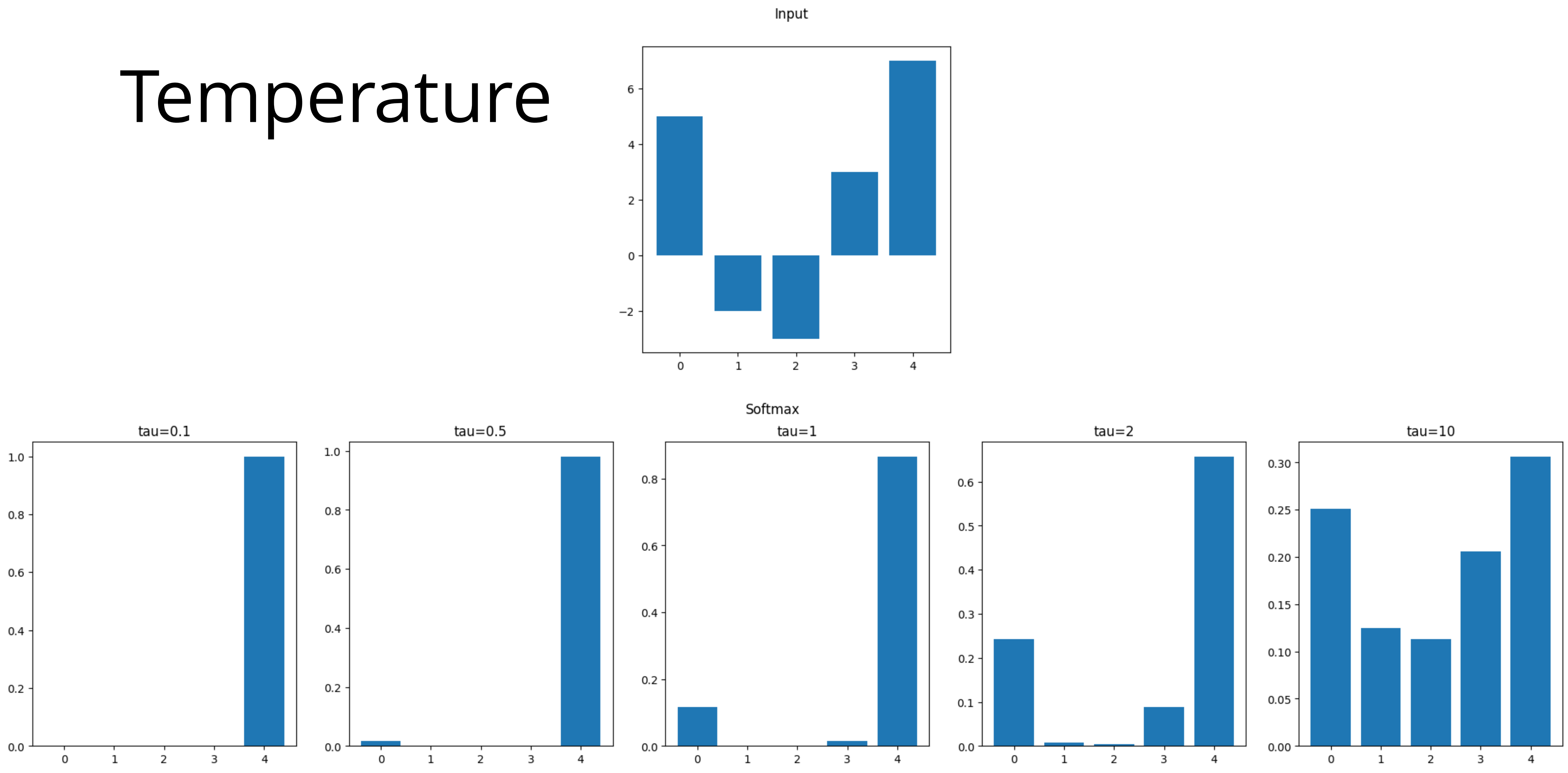
$$\mathrm{softmax}_k(\hat{Y}, \tau) = \frac{\exp \frac{\hat{Y}_k}{\tau}}{\sum_j \exp \frac{\hat{Y}_j}{\tau}} = \mathrm{softmax}_k(\frac{\hat{Y}}{\tau})$$

- Regulates the sharpness of the output distribution.
- Keeps relative ordering:

$$\mathrm{softmax}_i(\hat{Y}, \tau_1) < \mathrm{softmax}_j(\hat{Y}, \tau_1) \Rightarrow \mathrm{softmax}_i(\hat{Y}, \tau_2) < \mathrm{softmax}_j(\hat{Y}, \tau_2)$$

- As $\tau \to \infty$ softmax becomes a uniform distribution
- As $\tau \to 0$ softmax becomes argmax (in a vector representation).

Temperature



The temperature controls the *entropy* of the resulting probability distribution.

Cross-entropy loss

Soft-max classifier for K classes C_k:

$$p(C_k|x) = \mathrm{softmax}_k\, f(x) = \frac{\exp f_k(x)}{\sum_j \exp f_j(x)}$$

Loss function:

- Idea: maximise the probability of the ground-truth class.
- Minimise cross-entropy between ground-truth probability distribution (one-hot) and the predicted distribution.

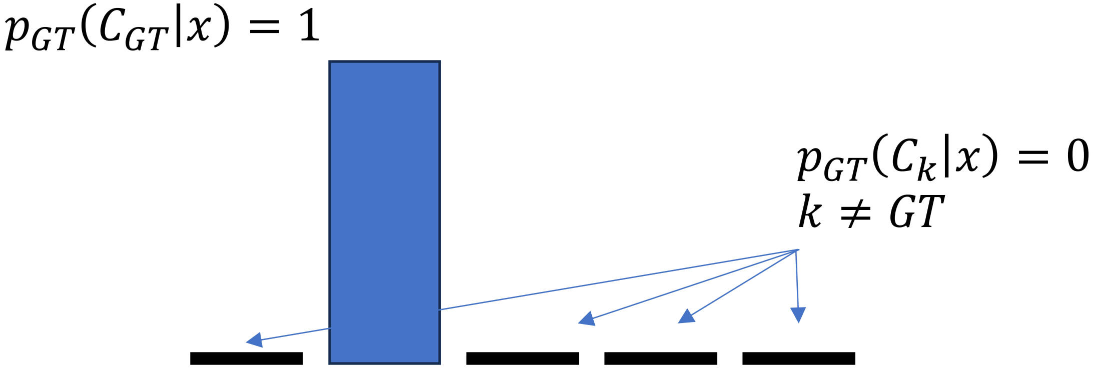
$$-\sum_j^K p_{GT}(C_j, x) \log\big(p(C_k|x)\big)$$

Cross-entropy loss

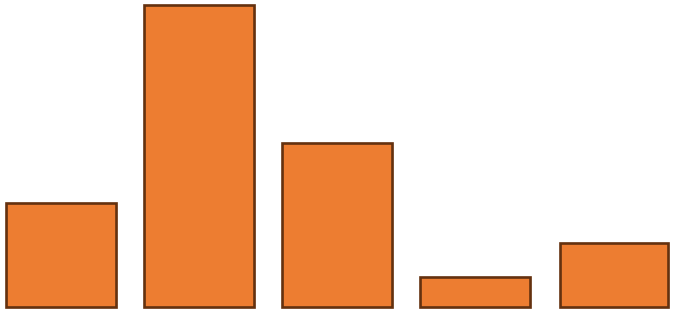
$$-\sum_{k}^{K} p_{GT}(C_k, x)\log\big(p(C_k|x)\big)$$

Since all $p_{GT}(C_k|x)$ are zero, except the target class C_{GT},i.e. $p_{GT}(C_{GT}, x) = 1$, this simplifies to

$$-\log\big(p(C_{GT}|x)\big) = -\log\frac{\exp f_{GT}(x)}{\sum_j \exp f_j(x)} = -f_{GT}(x) + \log\sum_{j}^{K}\exp\big(f_j(x)\big)$$



ground-truth distribution


predicted distribution

Cross-entropy loss

- The cross-entropy soft-max loss is differentiable.
- It is often used to train classification methods.
- There are some tricks to make it numerically stable.

Multi-Label Classification

- Multi-Class: each image has exactly one class of which there are many.
- Multi-Label: each image can have multiple classes.
- Classes might or might not be exclusive.

Image Classification in 2 Steps

1. Compute image embeddings.
2. Learn a classifier on the training set.

Many different approaches:

Image embeddings: FFT, BoW, HOG, Fisher Vectors, etc...

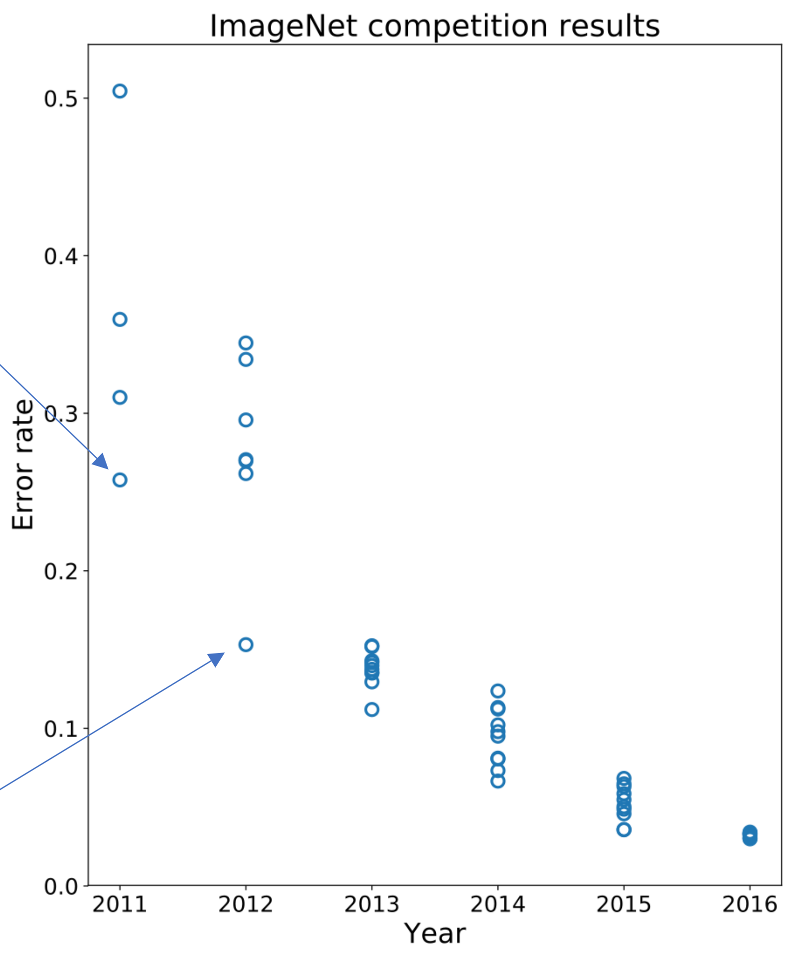
Classifiers: Linear Regression, SVMs, Kernel SVM, Random Forest, etc...

Deep Learning:

combine both steps and learn them simultaneously!

ImageNet Classification Challenge

SIFT-based



First *deep* method on this benchmark

A Krizhevsky, et al., 2012