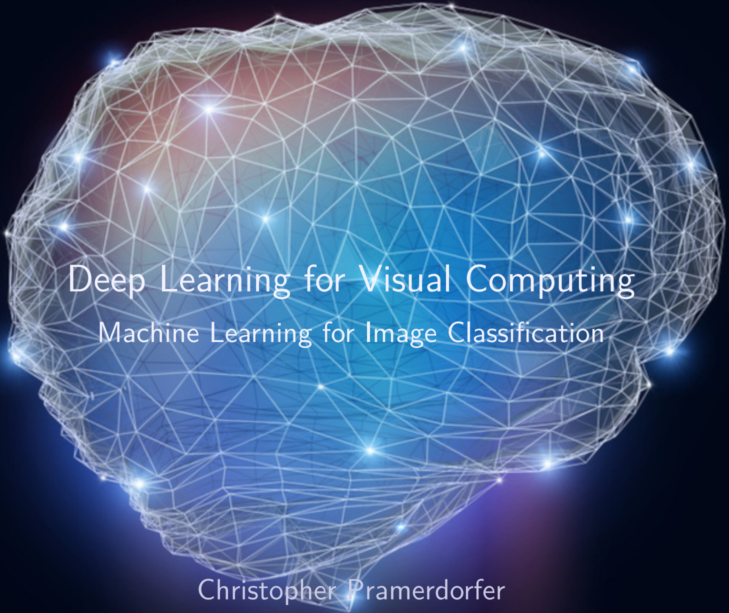


Deep Learning for Visual Computing

Machine Learning for Image Classification

Christopher Pramerdorfer

Computer Vision Lab, TU Wien

Image by Oracle

Topics

Machine Learning recap

Image classification using Nearest Neighbor classifiers

Validation sets and hyperparameter optimization

Motivation

We've seen that

- We humans are great image classifiers
- But we cannot formally describe how we do it
- So we can't write algorithms for reliable classification

This applies to most vision problems

- Reason we need Machine and Deep Learning

Motivation

Machine Learning (ML) algorithms are able to learn from data

- Algorithm performance improves with experience

In this framework we

- Select a suitable ML algorithm
- Collect data for algorithm training and evaluation
- Monitor the training progress

ML for Image Classification

Recall that we want to build an image classifier

- Should support the classes $\{$dog, cat$\}$
- Using the CIFAR-10 dataset (6000 images per class)

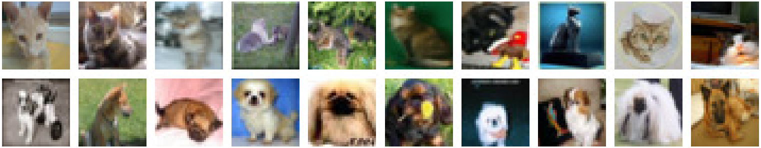


Image from cs.toronto.edu

ML for Image Classification

In this context

- We show (image, class) pairs to the chosen ML algorithm
- Algorithm learns to predict class of unseen samples

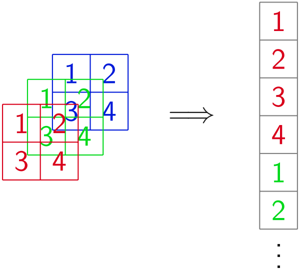
The information we are interested in is called label

- In our case the label value is *cat* or *dog*
- Finite number of label values, hence classification problem
- Labels used during training, hence supervised learning

ML for Image Classification
First Try

Represent images as points in D-dimensional input space

- Stack image rows/columns to obtain vector $\mathbf{x}$
- D is usually large (CIFAR-10 : $D = 32 \cdot 32 \cdot 3 = 3072$)



ML for Image Classification
First Try

Assuming $D = 2$ and three classes (colors)

- Can you think of a simple approach for classification?

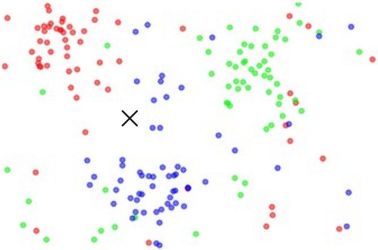


Image adapted from cs231n.github.io

ML for Image Classification
First Try

Nearest Neighbor classifier

- Compute distance from × to all training samples
- Assign label of closest sample

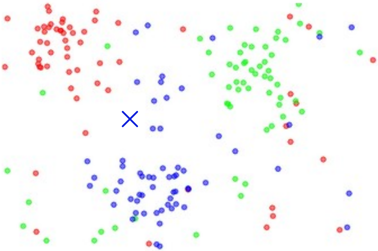


Image adapted from cs231n.github.io

ML for Image Classification
First Try

Need distance measure between two images $\mathbf{x}_1, \mathbf{x}_2$

L1 distance : $\sum_{d=1}^{D} \left| x_1^d - x_2^d \right|$

L2 (Euclidean) distance : $\sqrt{\sum_{d=1}^{D} \left(x_1^d - x_2^d \right)^2}$

ML for Image Classification
First Try

Resulting decision regions and decision boundaries

- Voronoi tessellation of input space

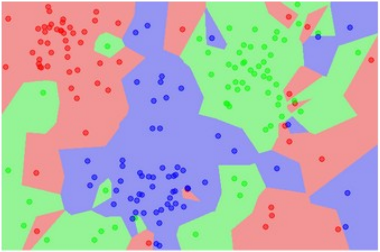
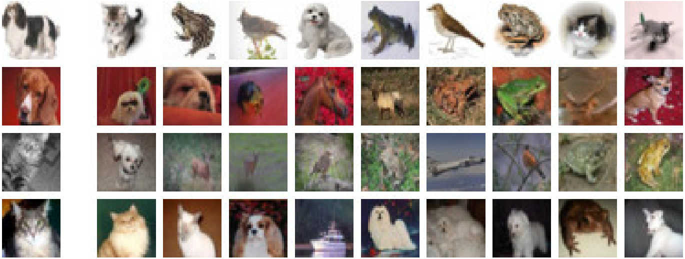


Image from cs231n.github.io

ML for Image Classification
First Try

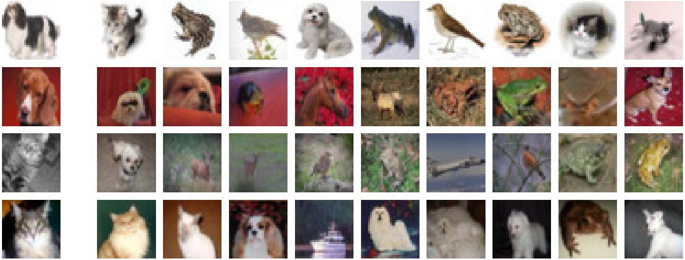
Performance (on full CIFAR-10 dataset)

- Obvious errors and challenges the classifier is not robust to
- But performance likely better than a “manual” method



ML for Image Classification
Second Try

How can we improve the performance?



ML for Image Classification
Second Try

k Nearest Neighbors classifier

- Find labels of k closest training samples
- Assign most frequent label

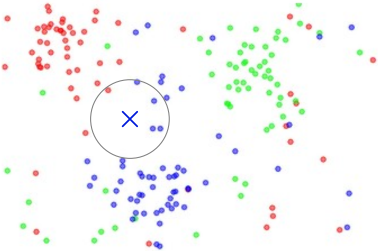


Image adapted from cs231n.github.io

ML for Image Classification
Second Try

Results in smoother decision boundaries

- Visualization suggest better performance on unseen samples

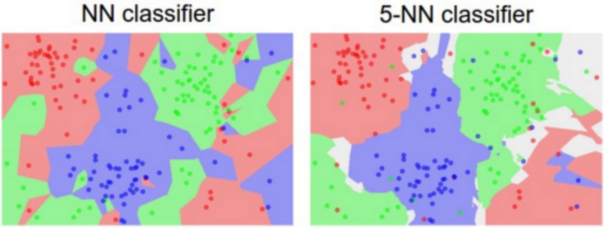


Image adapted from cs231n.github.io

ML for Image Classification
Second Try

We want to know how certain the classifier is

- Allows us to react differently on this basis

We can adapt the classifier accordingly

- Predict class scores $\mathbf{w} \in \mathbb{R}^C$ (C is number of classes)
- w_c is frequency of label c among k closest samples
- In example $\mathbf{w} = (1, 0, 4)$ assuming red, green, blue order

Class scores computed this way are not optimal

- Absolute values that depend on k
- Other classifiers will compute scores differently

We want $\mathbf{w}$ to be a valid probability mass function

- $w_c \geq 0$ for all c and $\sum_c w_c = 1$

Most popular function for this purpose is softmax

$$\text{softmax}_c(\mathbf{w}) = \frac{\exp(w_c)}{\sum_c \exp(w_c)}$$

We obtain $\text{softmax}((1, 0, 4)) \approx (0.05, 0.02, 0.93)$

- Largest value is emphasized, small ones suppressed
- Softmax is not scale invariant

ML for Image Classification
Second Try

How should we select k for cat vs. dog classification?

- Has a large impact on performance
- Different k work best depending on the data
- Cannot visualize the 3072-dimensional input space

k is a hyperparameter

- Set manually, controls algorithm behavior
- Chosen experimentally or based on domain knowledge

ML for Image Classification
Second Try

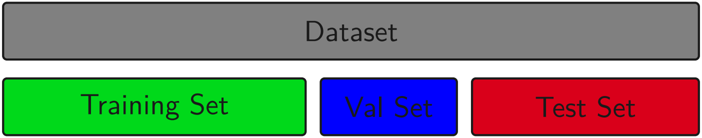
To find a good k for our problem, we

- Sample k from a reasonable range (what's reasonable?)
- Quantify the algorithm performance on a validation set
- Chose the k with the highest performance

Why not use the training or test sets?

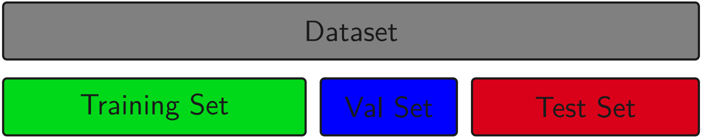
We thus need three disjoint datasets

- Training set for classifier training (duh)
- Validation set for hyperparameter selection
- Test set for final estimate of performance in practice



Test set used to estimate the performance in practice

- No longer valid if used during training/validation
- Use only once at the very end



ML for Image Classification
Second Try

Need suitable performance measure to quantify performance

Accuracy is popular for classification

- Let algorithm predict labels for dataset
- Compare predictions and true (ground truth) labels
- Accuracy is fraction of correctly classified samples

Measure $1 -$ accuracy is called error rate

We generally cannot test all hyperparameter combinations

- Testing one combination can take long
- Number blows up if we have several hyperparameters
- Large and/or continuous intervals

Use an approximative search strategy

- Grid search
- Random search

ML for Image Classification
Second Try

Given H hyperparameters with search intervals $I_1 \cdots I_H$

Grid search

- Sample uniformly from all I_h
- Test all combinations

Random search

- For j iterations, sample randomly from all I_h
- Test sample combination

ML for Image Classification
Second Try

Random search usually works better

- Wastes less time on unimportant hyperparameters

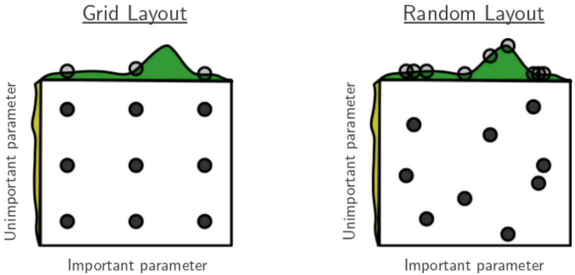


Image from [1]

ML for Image Classification
Second Try

k Nearest Neighbor classifier performs better but still not great

- More improvements in next lecture

Bibliography

[1] James Bergstra and Yoshua Bengio. *Random search for hyper-parameter optimization*. Journal of Machine Learning Research (2012).