

Deep Learning for Visual Computing

Machine Learning Basics

Christopher Pramerdorfer
Computer Vision Lab, TU Wien

Topics

Machine learning motivation

Image classification via machine learning

- ▶ Approach
- ▶ How algorithms learn
- ▶ Simple example : k nearest neighbors

Statistical learning theory

Hyperparameter selection

Motivation

Let's build an image classifier

- ▶ Should support the classes {dog, cat}
- ▶ Using the CIFAR10 dataset



Image from cs.toronto.edu

Motivation

How can we write an algorithm for this purpose?



Image from cs.toronto.edu

Motivation

We cannot!

- ▶ No obvious unique and reliable **features**
- ▶ Not clear how to represent and use them



Image from cs.toronto.edu

Motivation

We can classify images easily

But we cannot describe formally how to do so

- ▶ Thus the standard `if {} else {}` approach fails

This applies to most vision problems

Motivation

Machine Learning (ML) to the rescue!

ML algorithms are able to learn from data

Learning from data

- ▶ Performance of algorithm improves with experience
- ▶ Algorithm improves as it sees more cat and dog images

Motivation

ML is central in modern Computer Vision

DL is branch of ML

- ▶ Important ML concepts also apply to DL
- ▶ Why we review them in this lecture

Image Classification via ML

Approach

Show (image, label) pairs to ML algorithm

Algorithm somehow learns to predict labels for **unseen** samples

Finite number of different labels

- ▶ We have a **classification** problem (vs. **regression**)

We have samples and target labels

- ▶ Called **supervised learning** (vs. **unsupervised learning**)

Image Classification via ML

How Algorithms Learn

Images are points in D -dimensional space

- ▶ Stack image rows/columns to obtain vector \mathbf{x}
- ▶ D is usually large (CIFAR10 : $D = 32 \cdot 32 \cdot 3 = 3072$)

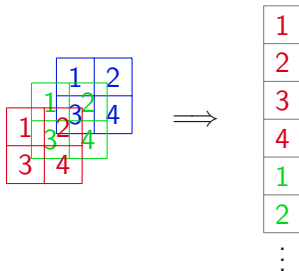


Image Classification via ML

How Algorithms Learn

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

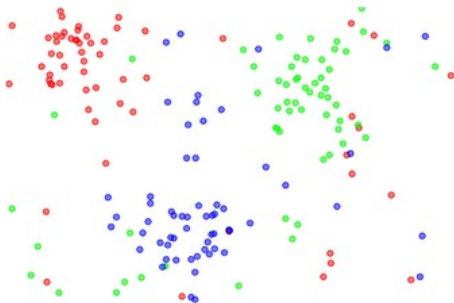


Image from cs231n.github.io

Image Classification via ML

How Algorithms Learn – Discriminative

Discriminative models learn decision boundaries

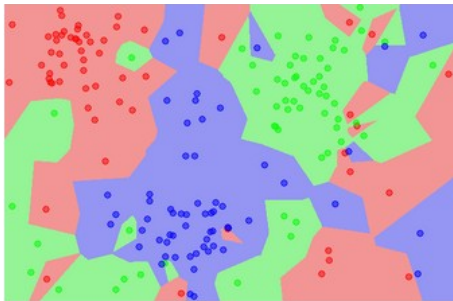


Image from cs231n.github.io

Image Classification via ML

How Algorithms Learn – Discriminative

Ideally such models are **probabilistic**

- ▶ Measure of uncertainty

Let w encode the class label, e.g. $w \in \{\text{cat}, \text{dog}\}$

Such models compute $\Pr(w|\mathbf{x})$

- ▶ **Conditional probability** of w given \mathbf{x}
- ▶ Probability for every class given image \mathbf{x}

Decision boundaries where $\arg \max_w \Pr(w|\mathbf{x})$ changes

Image Classification via ML

How Algorithms Learn – Generative

Generative models learn class-conditional densities $\Pr(\mathbf{x}|w)$

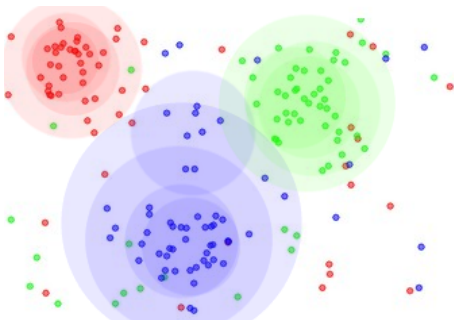


Image adapted from cs231n.github.io

Image Classification via ML

How Algorithms Learn – Generative

Classification from $\Pr(\mathbf{x}|w)$

- ▶ Compute $\Pr(\mathbf{x}|w) \Pr(w)$ for all w
- ▶ Assign class where product is maximal

Or to obtain $\Pr(w|\mathbf{x})$

- ▶ Compute $\Pr(\mathbf{x}|w) \Pr(w)$ for all w
- ▶ Divide each product by sum over all products

$\Pr(w)$ is fraction of training samples with class w

Image Classification via ML

How Algorithms Learn – Generative

This is **Bayes' rule** in action

$$\Pr(w|\mathbf{x}) = \frac{\Pr(\mathbf{x}|w) \Pr(w)}{\Pr(\mathbf{x})} = \frac{\Pr(\mathbf{x}|w) \Pr(w)}{\sum_w \Pr(\mathbf{x}, w)} = \frac{\Pr(\mathbf{x}|w) \Pr(w)}{\sum_w \Pr(\mathbf{x}|w) \Pr(w)}$$

Don't remember this stuff?

- ▶ See slide 21

Image Classification via ML

How Algorithms Learn – Generative

For example, assume that

- ▶ Classifier says $\Pr(\mathbf{x}|\text{cat}) = 0.03$ and $\Pr(\mathbf{x}|\text{dog}) = 0.022$
- ▶ We know that $\Pr(\text{cat}) = 0.4$ and $\Pr(\text{dog}) = 0.6$

We obtain

- ▶ $\Pr(\mathbf{x}|\text{cat}) \Pr(\text{cat}) = 0.0120$
- ▶ $\Pr(\mathbf{x}|\text{dog}) \Pr(\text{dog}) = 0.0132$
- ▶ $\Pr(\text{cat}|\mathbf{x}) = 0.48$ and $\Pr(\text{dog}|\mathbf{x}) = 0.52$

So we cannot be sure whether image shows cat or dog

Image Classification via ML

How Algorithms Learn

Discriminative models are more popular

- ▶ CNNs are discriminative and probabilistic

Different discriminate models/algorithms differ in

- ▶ Decision boundary properties (e.g. linear)
- ▶ How boundaries are computed

Image Classification via ML

How Algorithms Learn

Models vs. algorithms

- ▶ Model specifies boundary properties
- ▶ Algorithm specifies how to compute boundaries

Linear SVMs and logistic regression are both **linear models**

- ▶ But training and prediction algorithms differ

In practice, both terms often used interchangeably

Image Classification via ML

How Algorithms Learn

See “Computer Vision Models” book for details

- ▶ Available at computervisionmodels.com
- ▶ Highly recommended

Image Classification via ML

k Nearest Neighbors

Simple ML algorithm

Simply store the training set (images as vectors \mathbf{x})

Classify new image using labels of k nearest training samples

Image Classification via ML

k Nearest Neighbors

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

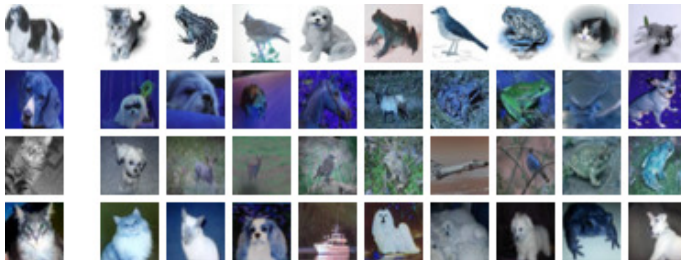
$$\text{L1 distance} : \sum_{d=1}^D |x_1^d - x_2^d|$$

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

Image Classification via ML

k Nearest Neighbors

Resulting cat vs. dog classifier



Algorithm Performance

Performance Measures

Looks like our classifier does not work so well

How can we quantify algorithm performance?

- ▶ Need a suitable **performance measure**

Algorithm Performance

Performance Measures

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**

Algorithm Performance

Training and Test Sets

What happens if we test on training data with $k = 1$?

Must test algorithm on data unseen during training!

- ▶ Want to assess its ability to **generalize** to unseen data
- ▶ Indicates how well it will perform in real world

Need two **disjoint** datasets

- ▶ **Training set** for learning
- ▶ **Test set** for assessing performance

Algorithm Performance

Training and Test Sets

How can this even work?

- ▶ We show the algorithm some data
- ▶ It should then perform well on different data

Proper training and test sets are highly correlated

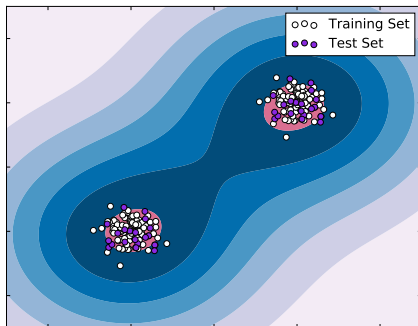
- ▶ Both reflect problem we want to solve
- ▶ Implies relationship between training and test accuracy

Algorithm Performance

Statistical Learning Theory

Samples of both sets generated by same underlying distribution

- ▶ So sets also have similar distribution



In our case, the underlying distribution

- ▶ Is distribution of *all* images that capture problem
- ▶ All possible cat and dog images (of certain size)

We clearly don't know how this distribution looks like

- ▶ But we can sample from this distribution
- ▶ By taking photos of cats and dogs

Algorithm Performance

Statistical Learning Theory

Can only collect so many samples

- ▶ Distributions more or less dissimilar
- ▶ Test error equal or greater than training error

Distributions of sets dissimilar to underlying distribution

- ▶ Problem known as [dataset bias](#)
- ▶ Test set performance generally upper bound

Algorithm Performance

Statistical Learning Theory

Two factors determine algorithm performance

- ▶ Ability to minimize training error
- ▶ Ability to minimize gap between training and test error

Algorithm Performance

Statistical Learning Theory

Related to two main challenges in ML

- ▶ Unable to reach low training error (**underfitting**)
- ▶ Gap between training and test error too large (**overfitting**)

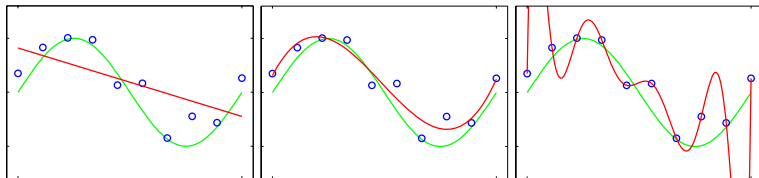


Image adapted from [1]

Must select suitable algorithm **capacity**

- ▶ Ability to compute complex decision boundaries

Lower capacity

- ▶ May struggle to fit the training set (higher **bias**)
- ▶ More prone to underfitting, less prone to overfitting

Higher capacity

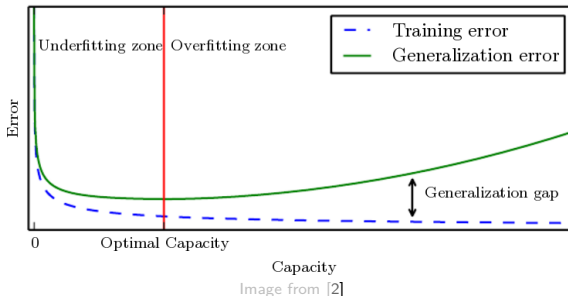
- ▶ May fit the training set too well (higher **variance**)
- ▶ Less prone to underfitting, more prone to overfitting

Algorithm Performance

Statistical Learning Theory

Test (generalization) error usually U-shaped function of capacity

- Decreases until some point, then increases again



Algorithm Performance

Statistical Learning Theory

Increasing training set size

- ▶ Increases performance as data reflect problem better
- ▶ Increases optimal capacity until adequate for problem

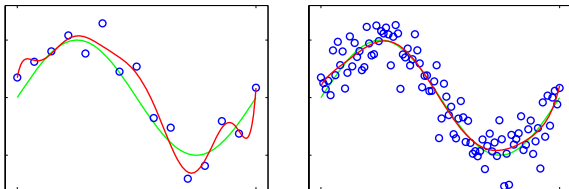


Image adapted from [1]

Hyperparameter Selection

k Nearest Neighbors

Capacity governed by k

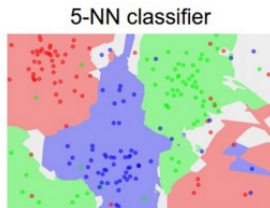
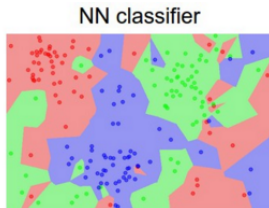


Image from cs231n.github.io

Hyperparameter Selection

Validation Sets

k is a **hyperparameter**

- ▶ Manually set, controls algorithm behavior

How should we set k ?

- ▶ Depending on data, different k work best
- ▶ Must find good choice experimentally

For this purpose we

- ▶ Use part of training set as **validation set**
- ▶ Use this set to see how different k perform

Hyperparameter Selection

Validation Sets

Validation set is split away from training set

- ▶ Never use test set during training!

Usually 10% to 30% of training data used for validation

- ▶ Depends on dataset size, number of hyperparameters

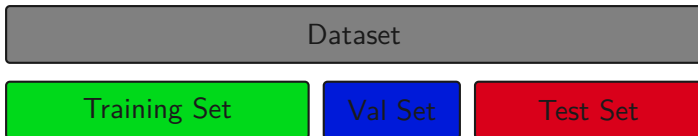
Alternatively we can use cross-validation

- ▶ Usually not done in DL (takes too long)

Hyperparameter Selection

Validation Sets

We thus need three disjoint datasets



Hyperparameter Selection

Search Strategies

We 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

Hyperparameter Selection

Search Strategies – Grid Search

For each hyperparameter

- ▶ Define search interval
- ▶ Sample from interval (often uniformly)

Test all sample combinations

(Repeat with smaller intervals based on previous results)

Hyperparameter Selection

Search Strategies – Random Search

Define search interval for each hyperparameter

For each trial

- ▶ Randomly sample value from each interval
- ▶ Test sample combination

Hyperparameter Selection

Search Strategies

Random search usually works better

- Wastes less time on unimportant hyperparameters

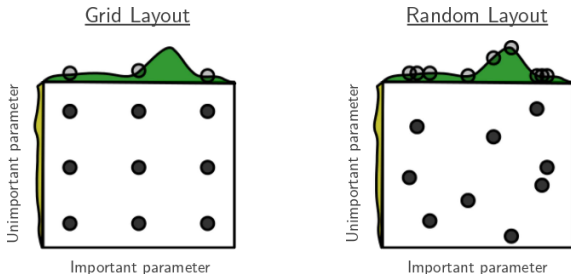


Image from [3]

Bibliography

- [1] C. Bishop, *Pattern Recognition and Machine Learning*. 2006.
- [2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. 2016. [Online]. Available: <http://www.deeplearningbook.org>.
- [3] J. Bergstra and Y. Bengio, *Random search for hyper-parameter optimization*, , 2012.