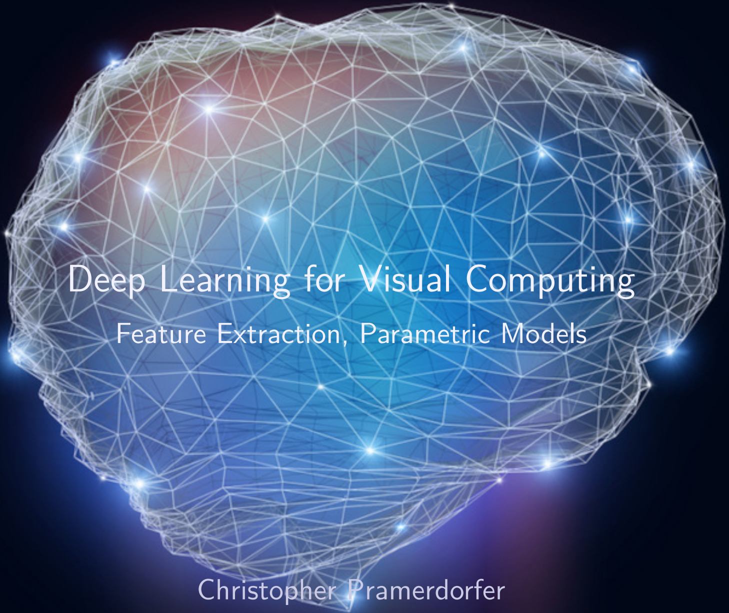


Deep Learning for Visual Computing

Feature Extraction, Parametric Models

Christopher Pramerdorfer

Computer Vision Lab, TU Wien

Image by Oracle

Topics

Feature extraction

- Low-level and high-level features

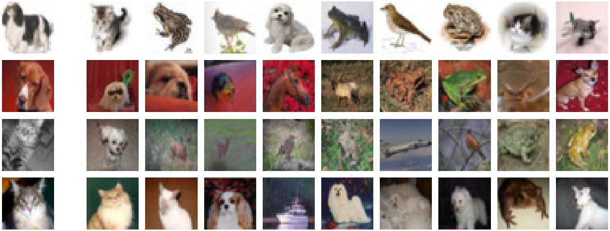
Parametric models

- Linear classifiers

Feature Extraction
Motivation

Our k Nearest Neighbors classifiers still perform poorly

- Classification based on overall image similarity
- On full CIFAR-10 dataset, test accuracy is about 40%



Mainly because we use raw images as feature vectors $\mathbf{x}$

But the classifiers used have no understanding of images

- Hence the features are not very useful

Also D is large so the curse of dimensionality strikes

- Input space is sparsely occupied by training samples
- Distance measures become unintuitive

Raw pixel values are poor features

- A feature is certain property of data

Goal of feature extraction

- Extract discriminative features from images
- Discriminative features help distinguish between classes

What are good features of natural images?

Feature Extraction
High-Level Features

Ideal features are task-specific

- Presence of pointy ears, long furry tails, whiskers
- Obviously very discriminative

Such features are called high-level features

- Tell us something about the world depicted

But we again face the same dilemma

- We humans use ("extract") these features
- But we cannot write such high-level feature extractors

Feature Extraction
High-Level Features

Deep Learning (DL) models are able to learn such features

- Main reason they are so powerful

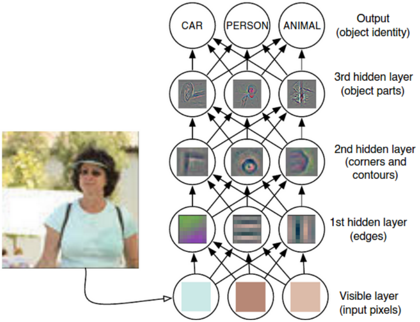


Image from [1]

For now consider generally useful features we can design manually

- Properties about the image itself (low-level)

Brightness changes at certain scale and/or orientation are popular

- Invariant to additive luminance changes
- Response at (certain) object borders, textured regions

This is where Image Processing comes in

Respond to changes at certain frequency and orientation

- Modeled after operations applied to visual data in brain
- DL models learn similar filters

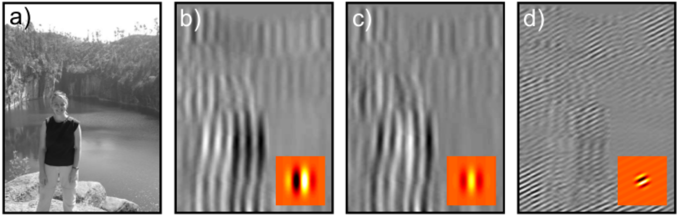


Image from [2]

More compact representation of distribution of brightness changes

1. Compute gradient magnitude and orientation
2. Divide image into overlapping cells
3. Compute histogram of quantized orientations in every cell
4. Combine cells to blocks, concatenate histograms
5. Normalize blocks, concatenate to single vector

See [2] for details

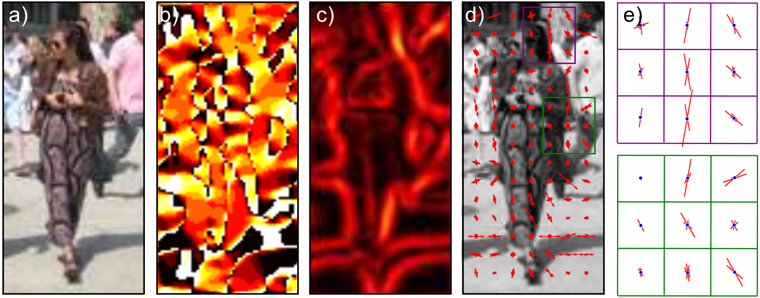


Image from [2]

Desirable to reduce the dimensionality D of feature vectors

- Combat curse of dimensionality
- Facilitate training and inference

Goal is to preserve as much information as possible

- Possible because images of certain type occupy a manifold
- Small subset of input space

PCA is popular for this purpose

Feature Extraction
Low-Level Features – Dimensionality Reduction

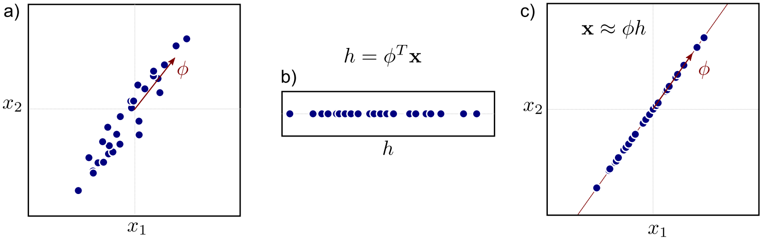


Image from [2]

Traditional Image Classification Pipeline

This is the traditional image classification pipeline

1. Extract low-level feature vectors
2. Perform dimensionality reduction
3. Process using generic ML algorithm like SVM

Standard pipeline until around five years ago

- Now replaced by Deep Learning pipeline

Traditional Image Classification Pipeline

HoG features improve CIFAR-10 performance by about 10%

- Still not where we want to be

Reasons

- k Nearest Neighbors is a simple classifier
- HoG features are low-level

k Nearest Neighbors classifier has several limitations

- Must keep all training samples
- Prediction is slow
- Not particularly powerful

Parametric models overcome these limitations

Let $\mathbf{x} \in \mathbb{R}^D$ be input and $\mathbf{w} \in \mathbb{R}^T$ be output

- Last time we used C instead of T

A model describes family of functions from $\mathbf{x}$ to $\mathbf{w}$

- Particular function $f : \mathbf{x} \mapsto \mathbf{w}$ learned during training

Model defines the hypothesis space of a ML algorithm

- Set of functions allowed as solution
- Extending family increases the model capacity (flexibility)

In parametric models f depends on parameters $\boldsymbol{\theta}$

- We write $\mathbf{w} = f(\mathbf{x}; \boldsymbol{\theta})$
- Training entails finding good parameters
- Training set can be discarded after training

DL models can have millions of parameters

The most basic example are linear models

Hypothesis space comprises linear functions from $\mathbb{R}^D$ to $\mathbb{R}^T$

- Formally $\mathbf{w} = f(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ with $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$

Parameters

- $\mathbf{W} \in \mathbb{R}^{T \times D}$ is called weight matrix
- $\mathbf{b} \in \mathbb{R}^T$ is called bias vector

Linear models are important components of DL classifiers

We can regard such classifiers as a combination of

- An adaptive extractor of high-level features
- Followed by a linear classifier

Why we cover linear classifiers at this point

A (unsuccessful) example prediction

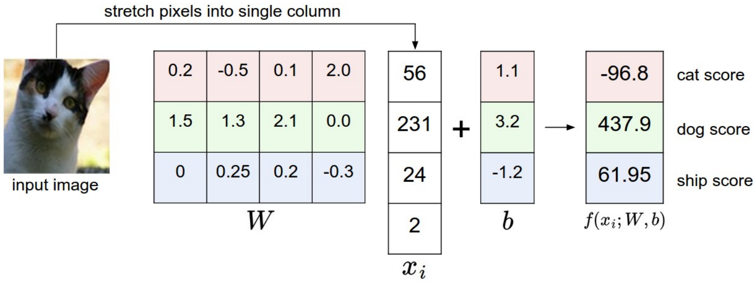


Image from cs231n.github.io

Linear Models for Classification
Example

Can think of T independent linear classifiers

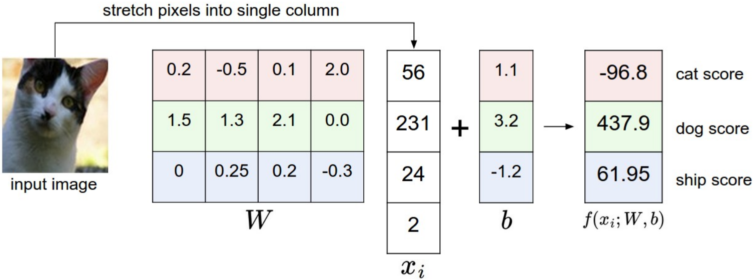


Image from cs231n.github.io

T hyperplanes as decision boundaries in $\mathbb{R}^D$

- Weights define orientation, bias defines offset from 0

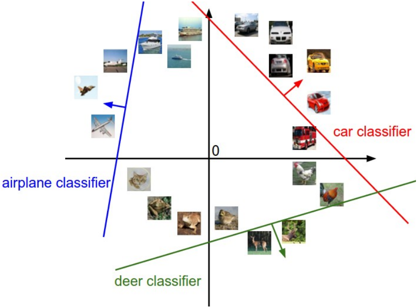


Image from cs231n.github.io

Linear Models for Classification

Geometric Interpretation

Hyperplane c should answer "is $\mathbf{x}$ of class c?"

- Want $\mathbf{x}$ on positive side of hyperplane c if of class c

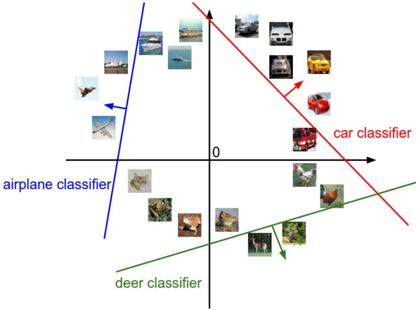


Image from cs231n.github.io

$\mathbf{x}$ is on positive side if $w_c = \mathbf{w}_c\mathbf{x} + b_c \geq 0$

- $\mathbf{w}_c$ is row c of $\mathbf{W}$

Signed distance to plane is $w_c/\|\mathbf{w}_c\|$

- Hence class score increases with distance/confidence

Can again apply softmax function to get probabilistic scores

Linear Models for Classification
Geometric Interpretation

Goal of training: make hyperplanes always answer correctly

- Only possible if classes are linearly separable

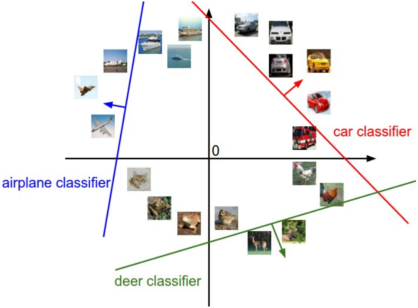


Image from cs231n.github.io

Hyperplanes do not work together

- Each hyperplane is an independent binary classifier

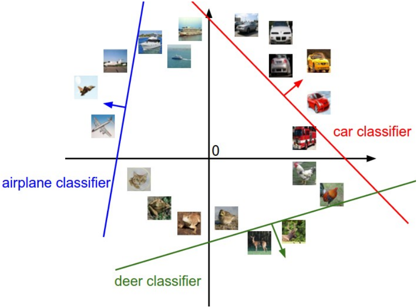


Image from cs231n.github.io

T learned templates that are matched with input images

- Each $\mathbf{w}_c$ encodes a template
- Matching using inner product $\mathbf{w}_c\,\mathbf{x}$ (plus b_c)
- Class score increases with similarity of image to template

Image from cs231n.github.io

Most templates have clear interpretation

- Horse template shows something horse-like
- Most cars training data seem to be red
- Background is (again) very dominant (sky, grass, water)

Image from cs231n.github.io

Linear classifiers cannot properly model intraclass variation

- Templates merge modes of variation
- What about blue cars, planes on ground, gray horses?

Image from cs231n.github.io

Efficient inference

- Single matrix-vector multiplication

Number of parameters governed by D and T

- $\mathbf{W} \in \mathbb{R}^{T \times D}$ and $\mathbf{b} \in \mathbb{R}^{T}$
- CIFAR-10 : $T = 10$ and $D = 3072$, so 30730 parameters

Special case is $T = 2$, needs only one decision boundary

- Like in our cats vs. dogs example

Can apply the bias trick to simplify f to $f(\mathbf{x}) = \mathbf{W}\mathbf{x}$

- Append $\mathbf{b}$ to $\mathbf{W}$, append 1 to $\mathbf{x}$

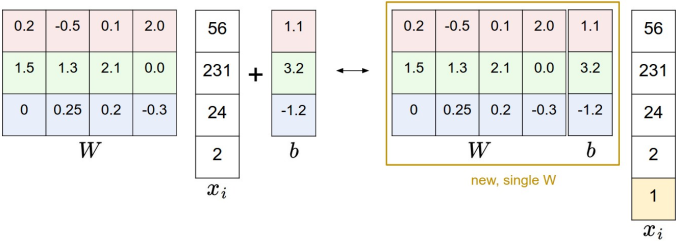


Image from cs231n.github.io

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 2016.

[2] Prince. *Computer Vision Models*. 2012.