

Feature Extraction

ML: Data + Feature + Model
different apps
different feature
general



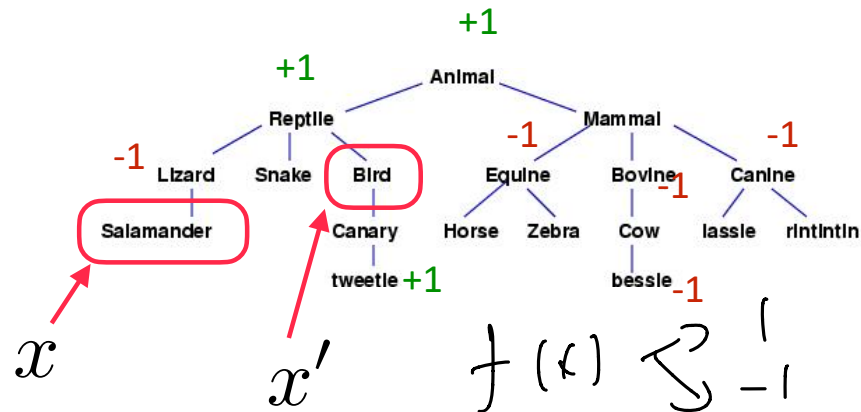
Feature extraction

Data comes in all forms:

Real, continuous features $x \in \mathbb{R}^d$ $x = [0.1, 4.0, 4.3, \dots, 2.5]^\top$

Categorical data $x = [\text{Red}, 98105, \text{Finished basement}, \dots, 2.5]^\top$

Structured data





Given tree and labels are known at some nodes,
how do we predict unknown labels?

Feature extraction

Data comes in all forms:

Text data





<http://www.ratebeer.com/beer/two-hearted-ale/>

3.8 AROMA 8/10 APPEARANCE 4/5 TASTE 8/10 PALATE 3/5 OVERALL 15/20
fonetan (25678) - Vestjylland, DENMARK - JAN 18, 2009

Bottle 355ml.
Clear light to medium yellow orange color with a average, frothy, good lacing, fully lasting, off-white head. Aroma is moderate to heavy malty, moderate to heavy hoppy, perfume, grapefruit, orange shell, soap. Flavor is moderate to heavy sweet and bitter with a average to long duration. Body is medium, texture is oily, carbonation is soft. [250908]

4 AROMA 8/10 APPEARANCE 4/5 TASTE 7/10 PALATE 4/5 OVERALL 17/20
Ungstrup (24358) - Oamaru, NEW ZEALAND - MAR 31, 2005

An orange beer with a huge off-white head. The aroma is sweet and very freshly hoppy with notes of hop oils - very powerful aroma. The flavor is sweet and quite hoppy, that gives flavors of oranges, flowers as well as hints of grapefruit. Very refreshing yet with a powerful body.

Image data



Audio data



Time-series data



Feature Extraction given real-valued data

- given $x \in \mathbb{R}^d$, improve the quality



Feature extraction - real vectors

Real, continuous features $x \in \mathbb{R}^d$ $x = [0.1, 4.0, 4.3, \dots, 2.5]^\top$
 $f(x) \rightarrow y$

Strategies if many features are **uninformative**?

- feature selection (LASSO)
- no useful feature \rightarrow collect new data

Feature extraction - real vectors

Real, continuous features $x \in \mathbb{R}^d$ $x = [0.1, 4.0, \underbrace{4.3}_{\text{small}}, \dots, \underbrace{2.5}_{\text{big}}]^\top$

Strategies if many features are **incomparable**?

→ standardization

Feature extraction - real vectors

Real, continuous features $x \in \mathbb{R}^d$ $x = [0.1, 4.0, 4.3, \dots, 2.5]^\top$

Strategies if many features are **superfluous** or correlated with each other?

- feature selection
- de-correlation (PCA)
→ independent features

Feature extraction - real vectors

Real, continuous features

$$x \in \mathbb{R}^d \quad x = [0.1, 4.0, 4.3, \dots, 2.5]^\top$$

x_1, \dots, x_n (4-d) $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$, $\text{std}(x_i)$

Pre-processing pipeline:

1. Standardize data (de-mean, divide by standard deviation)
2. Project down to lower dimensional representation using PCA
3. Apply exact transformation to Train and Test.

} are always useful
→ on validation set

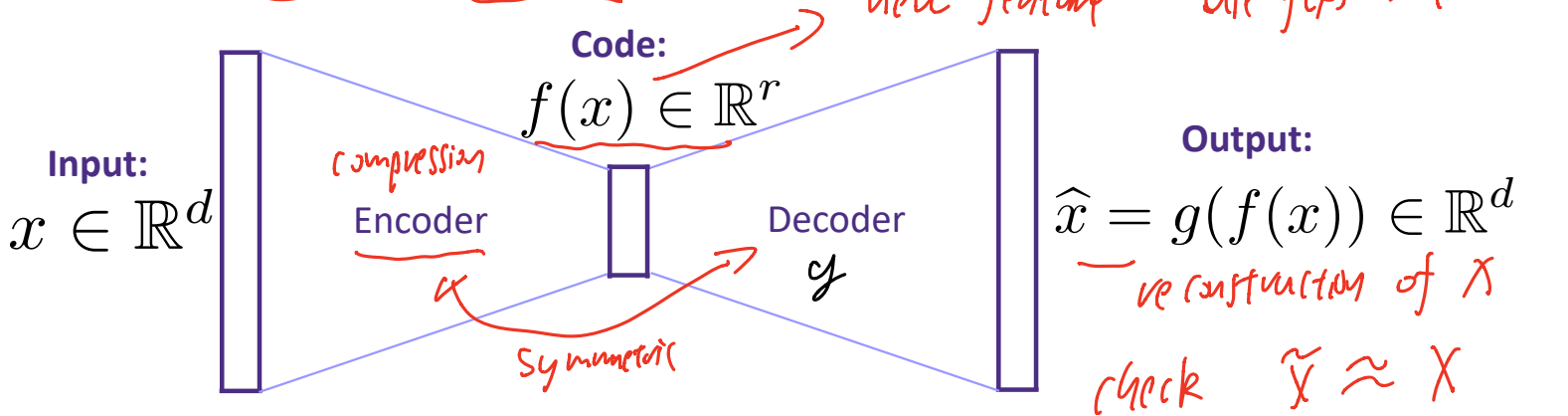
Test data: subtract training mean $\frac{x^{\text{test}} - \bar{x}_{\text{train}}}{\text{std}(x_{\text{train}})}$
divide by training std
project with PCA computed by training

Autoencoders

non-linear PCA

$$r < d$$

Find a low dimensional representation for your data by predicting your data



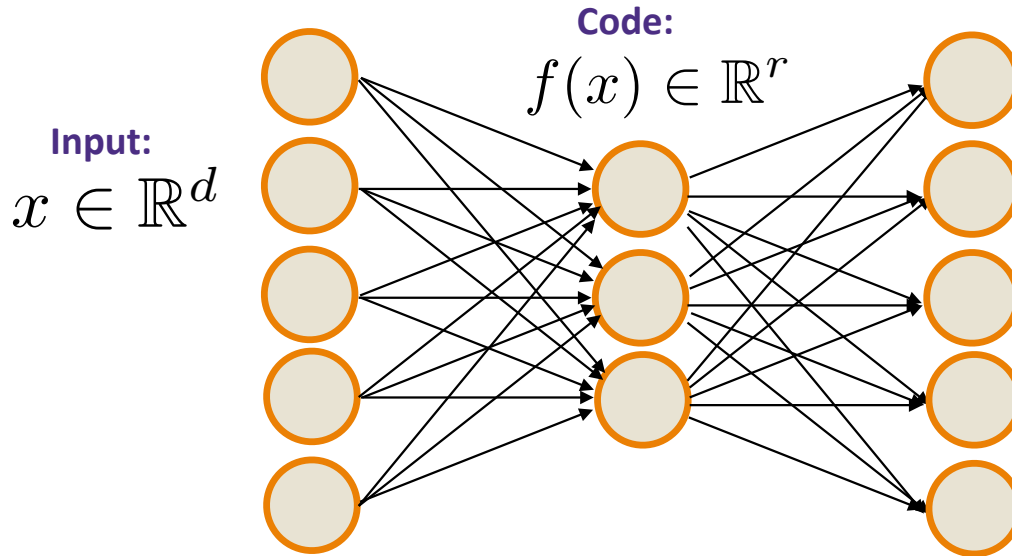
$$\underset{f,g}{\text{minimize}} \sum_{i=1}^n \|x_i - g(f(x_i))\|_2^2$$

f : neural net
 g : neural net

$$f: \mathbb{R}^d \rightarrow \mathbb{R}^r \quad g: \mathbb{R}^r \rightarrow \mathbb{R}^d$$

↑
non-linear

Autoencoders



$$\text{minimize}_{f,g} \sum_{i=1}^n \|x_i - g(f(x_i))\|_2^2$$

$$A \in \mathbb{R}^{r \times d}$$

$$B: \mathbb{R}^{d \times r}$$

$$\min_{B, A} \|BAX - Y\|^2$$

solution: $A, B = A^+$

What if $f(X) = AX$ and $g(y) = By$?

$$\hat{X} = \underbrace{BA}_{\text{low-rank projection}} X \Leftrightarrow \text{rank}(A)$$

Feature Extraction given categorical data



Feature extraction - categorical

Categorical data $x = [\text{Red}, 98105, \text{Finished basement}, \dots, 2.5]^\top$

Many machine learning algorithms (e.g., linear predictors) require real valued-vectors to make predictions.

And we want those real-valued numbers to be **correlated with the label**.

Feature extraction - categorical

Categorical data $x = [\text{Red}, 98105, \text{Finished basement}, \dots, 2.5]^T$

Many machine learning algorithms (e.g., linear predictors) require **real valued-vectors** to make predictions.

And we want those real-valued numbers to be **correlated with the label**.

One-hot encoding: Assign canonical vector to each categorical variable

color $\in \{\text{red}, \text{green}, \text{blue}\}$

unstructured
encoding

$$\text{red} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{green} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\text{blue} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

another encoding

$$\begin{cases} \text{red} = 1 \\ \text{green} = 2 \\ \text{blue} = 3 \end{cases}$$

\rightarrow

$$\text{red} + \text{green} = \text{blue}$$

\times

Feature extraction - categorical

Categorical data $x = [\text{Red}, \underline{98105}, \text{Finished basement}, \dots, 2.5]^T$

Many machine learning algorithms (e.g., linear predictors) require **real valued-vectors** to make predictions.

And we want those real-valued numbers to be **correlated with the label**.

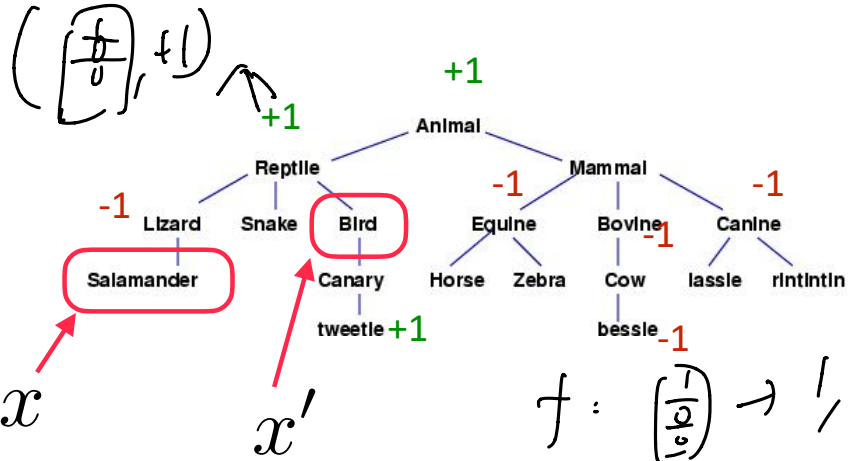
Zip codes are also categorical. Is one-hot encoding appropriate?

zip code = 98105 # of zip > 40,000

- 40,000 vector just for zip code
- (compress)
98105 / 98106 → same

Feature extraction - structured

Structured data



$$d(x, x') \rightarrow \mathbb{R}^T$$

x

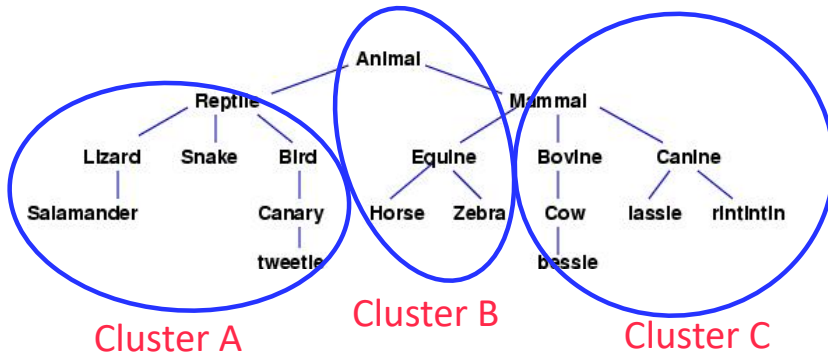
x'

$$f: \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \rightarrow \text{!} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \rightarrow \text{!}$$

Trees define a distance between any two nodes (length of path connecting them)

$$\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \rightarrow -1$$

Given distances, you can assign each node to a cluster



Cluster A

Cluster B

Cluster C

Then one-hot encode:

$$\text{cluster} \in \{A, B, C\}$$

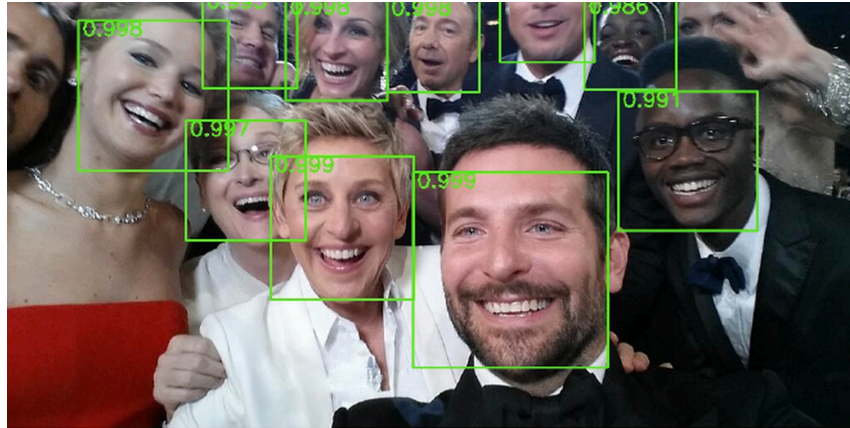
$$\text{snake} \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\text{cow} \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Feature extraction given Image data



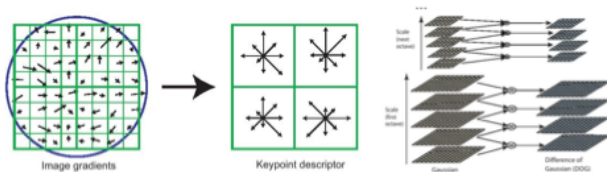
Computer Vision



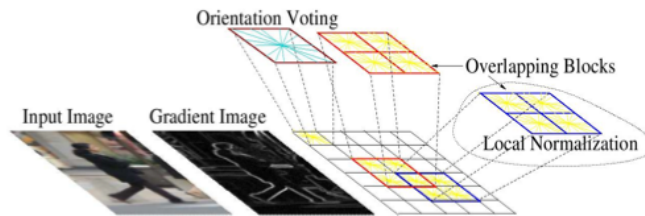
Find a feature vector for the image:

- Recognition
- Identification
- Detection
- Image classification
- etc...

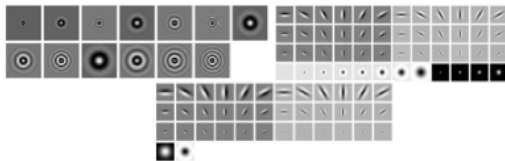
Some hand-created image features



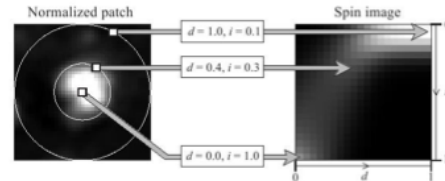
SIFT



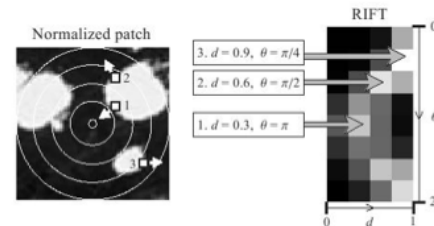
HoG



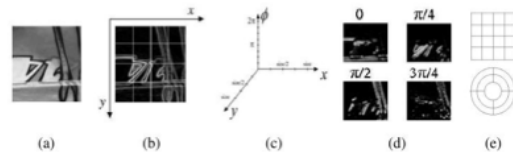
Texon



Spin Image

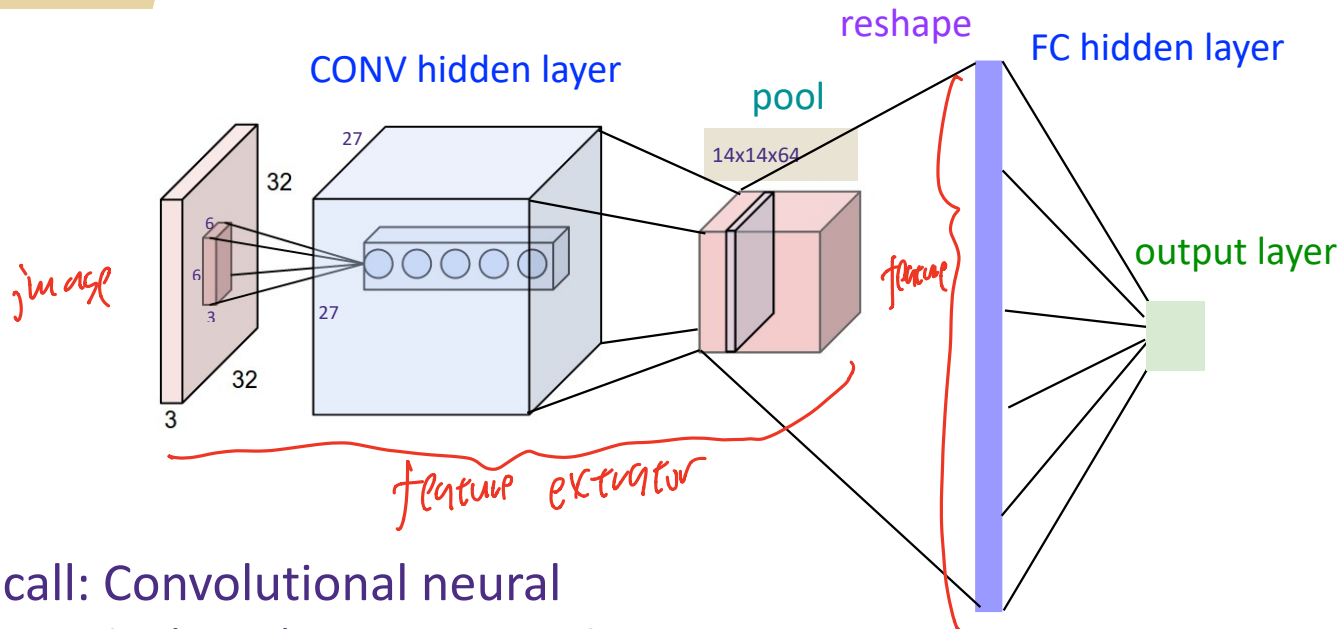


RIFT



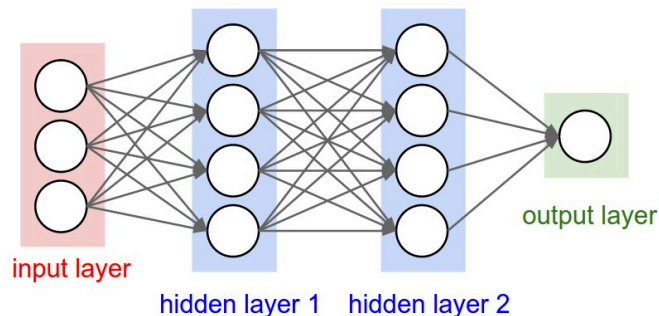
GLOH

Learning Features with Convolutional Networks



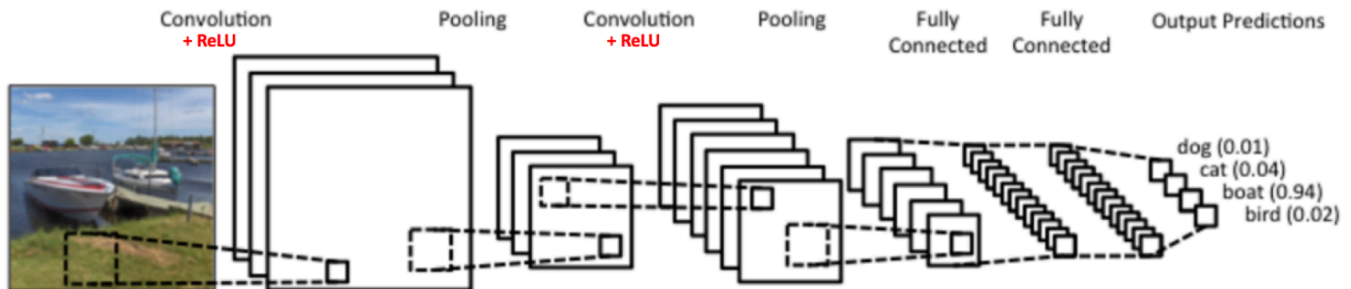
Recall: Convolutional neural networks (CNN) are just regular fully connected (FC) neural networks with some connections removed.

Train with SGD!





Real example network: LeNet



Real networks

Residual Network of
[HeZhangRenSun'15]

Wiki

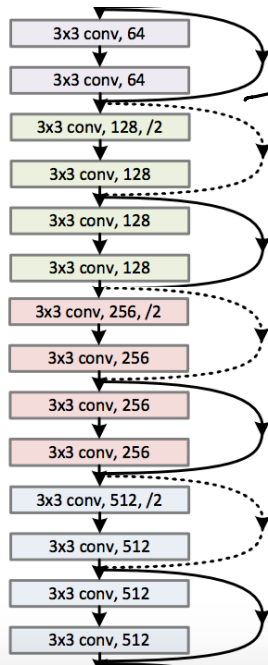
$28 \times 28 \times 3$



Reduce spatial
dimension

Reduce spatial
dimension

Reduce spatial
dimension



$28 \times 28 \times 64$
 $\rightarrow 14 \times 14 \times 128$

feature
↓
linear classifier

feature extractor
train on ImageNet
(1M)

new data
use feature extractor
 \rightarrow vector
vector linear layer

may train feature
extractor (fine-tune)