Machine Learning for Image Processing and Synthesis

Julie Digne

LIRIS - CNRS
Outline

1. Introduction
2. General Formulation
3. Support Vector Machine
4. On Trees and Forests
5. Boosting
7. Neural nets
8. Generative problems
9. Generative Adversarial Networks (GAN)
Classical Vision Algorithms

- Line detection:

Around 10000 to 30000 different objects to model: a model for each of them. Not doable in practice.
Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
  Example: two close-by ellipsoids
Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
  Example: two close-by ellipsoids
- Airplane recognition: devise a model for an airplane.
Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
  Example: two close-by ellipsoids
- Airplane recognition: devise a model for an airplane.
  Example one rectangle and two triangles?
Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
  Example: two close-by ellipsoids
- Airplane recognition: devise a model for an airplane.
  Example one rectangle and two triangles?
- How about when it’s more complicated?
Classical Vision Algorithms

- Line detection: RANSAC/Hough for line parameter estimation.
- Cat recognition: devise a model for a cat.
  Example: two close-by ellipsoids
- Airplane recognition: devise a model for an airplane.
  Example: one rectangle and two triangles?
- How about when it’s more complicated?

Problem

Around 10000 to 30000 different objects to model: a model for each of them **Not doable in practice.**
Machine learning and vision

- Recognition/detection

- Synthesis
Machine learning and vision

- **Recognition/detection**
  - Recognize objects in an image/video.
Machine learning and vision

- **Recognition/detection**
  - Recognize objects in an image/video.
  - Locate an object in an image/video
Machine learning and vision

- **Recognition/detection**
  - Recognize objects in an image/video.
  - Locate an object in an image/video
  - Recognize a behavior/an emotion in a video
Machine learning and vision

- **Recognition/detection**
  - Recognize objects in an image/video.
  - Locate an object in an image/video.
  - Recognize a behavior/an emotion in a video.

- **Synthesis**
Machine learning and vision

- **Recognition/detection**
  - Recognize objects in an image/video.
  - Locate an object in an image/video
  - Recognize a behavior/an emotion in a video

- **Synthesis**
  - Generate an image that looks like a set of examples
Machine learning and vision

- **Recognition/detection**
  - Recognize objects in an image/video.
  - Locate an object in an image/video
  - Recognize a behavior/an emotion in a video

- **Synthesis**
  - Generate an image that looks like a set of examples
  - Generate an image from a sketch given by a user.
Supervised Learning a set of data \((x_i)_i\) and associated labels (ex: cat, car, house...) \((l_i)_i\), learn a function \(\hat{f}\) such that \(f(x_i) = l_i\).
Supervised and Unsupervised Learning

- **Supervised Learning** a set of data \((x_i)i\) and associated labels (ex: cat, car, house...) \((l_i)i\), learn a function \(\hat{f}\) such that \(f(x_i) = l_i\).

- **Unsupervised Learning** a set of data \((x_i)i\) without any label and learns from similarities between data.
Some examples from previous classes

- Meanshift
Some examples from previous classes

- Meanshift
- K-means
Some examples from previous classes

- Meanshift
- K-means
- Expectation-Maximization
Some examples from previous classes

- Meanshift
- K-means
- Expectation-Maximization

Grouping problems

Unsupervised learning: no label provided for learning the classes.
Some examples from previous classes

- Meanshift
- K-means
- Expectation-Maximization

Grouping problems

Unsupervised learning: no label provided for learning the classes.

- Focus on supervised learning for today.
Outline

1. Introduction
2. General Formulation
3. Support Vector Machine
4. On Trees and Forests
5. Boosting
7. Neural nets
8. Generative problems
9. Generative Adversarial Networks (GAN)
General setup of a supervised machine learning problem

- Data: split into:
  - Training data
  - Test data
  - Evaluation data

- Given data and labels \((x_i, l_i)_i\), find \(f\) minimizing the *objective* function:

\[
\sum_i (f(x_i) - l_i)^2
\]
General setup of a supervised machine learning problem

- Data: split into:
  - Training data
  - Test data
  - Evaluation data
- Given data and labels \((x_i, l_i)_i\), find \(f\) minimizing the objective function:
  \[
  \sum_i (f(x_i) - l_i)^2
  \]
- Several objective functions exist (also called loss)
Underfitting and Overfitting

Under-fitting
(too simple to explain the variance)

Appropriate-fitting

Over-fitting
(forcefitting -- too good to be true)
Underfitting and Overfitting

Under-fitting
(too simple to explain the variance)

Appropriate-fitting

Over-fitting
(forcefiting -- too good to be true)
Precision and Recall

- **Precision**: how accurate is the classifier in detecting a positive example and not misclassifying.

\[
\text{Precision} = \frac{\#\text{True positives}}{\#\text{True positives} + \#\text{False positives}}
\]

- **Recall**: how accurate is the classifier in correctly detecting a positive example.

\[
\text{Recall} = \frac{\#\text{True positives}}{\#\text{positive examples}}
\]

Precision and recall curves are usually drawn with respect to the number of training iterations.

**Other indicators**
Bias, variance, confusion matrix...
Outline

1 Introduction
2 General Formulation
3 Support Vector Machine
4 On Trees and Forests
5 Boosting
6 Beyond classification: Dictionary Learning
7 Neural nets
8 Generative problems
9 Generative Adversarial Networks (GAN)
Support Vector Machine

Large margin binary classifier

Find an hyperplane separating the two classes maximizing the *margin*. 

![Diagram of Support Vector Machine with an hyperplane separating two classes](Image By Lahmam - Own work, CC BY-SA 4.0, wikipedia.)
Support Vector Machine

- Works well when the classes are linearly separable. What if it’s not the case?
Support Vector Machine

- Works well when the classes are linearly separable. What if it’s not the case?

Kernel trick

Find a function $\Phi$ such that the two classes of $(\phi(x_i), l_i)_i$ are linearly separable.

Image David Filliat (ENSTA).
Support Vector Machine

- Works well when the classes are linearly separable. What if it’s not the case?

**Kernel trick**

Find a function $\Phi$ such that the two classes of $(\phi(x_i), l_i); i$ are linearly separable.

Image David Filliat (ENSTA).

- Problem: how do we design $\Phi$? **Manually**
Support Vector Machine

- Works well when the classes are linearly separable. What if it’s not the case?

**Kernel trick**

Find a function $\Phi$ such that the two classes of $(\phi(x_i), l_i); i$ are linearly separable.

![Diagram of input space and feature space transformation](Image David Filliat (ENSTA).)

- Problem: how do we design $\Phi$? Manually or that's where Deep Learning methods come in handy.
Optimization

- Equation of the separating hyperplane: $w^T x + b = 0$
- If $w^T x_i + b > 0$ then $l_i = 1$, and if $w^T x_i + b < 0$ then $l_i = -1$,
- Decision function: $f(x) = \text{sign}(w^T x + b)$.

**Maximal margin**

Maximize the distance between hyperplanes $w^T x + b = \pm 1$. Decision function: $l_i = 1$ if $w^T x + b \geq 1$, $l_i = -1$ if $w^T x + b \leq 1$.

What is the size of the margin between the two hyperplanes?
Optimization problem

\[
\text{Minimize}_{w, b} \frac{1}{2} w^T w
\]

subject to \( \forall i, l_i (w^T x_i + b) \geq 1 \)
Optimization problem

\[
\text{Minimize}_{w,b} \frac{1}{2} w^T w
\]

subject to \( \forall i, l_i (w^T x_i + b) \geq 1 \)

Allow for some training errors: samples that violates the margin condition

Reformulation

\[
\text{Minimize}_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i
\]

subject to \( \forall i, l_i (w^T x_i + b) \geq 1 - \xi_i \) and \( \forall i, \xi_i \geq 0 \)
Optimization (continued)

- Notice that constraints are equivalent to $\xi_i = \max(0, 1 - l_i(w^T x_i + b))$
Optimization (continued)

- Notice that constraints are equivalent to $\xi_i = \max(0, 1 - l_i (w^T x_i + b))$
- set $x_i = [x_i; 1]$ and $w = [w; b]$ to simplify.
Optimization (continued)

- Notice that constraints are equivalent to $\xi_i = \max(0, 1 - l_i(w^T x_i + b))$
- set $x_i = [x_i; 1]$ and $w = [w; b]$ to simplify.

**Reformulation**

$$\text{Minimize}_w J(w) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \max(0, 1 - l_i(w^T x_i))$$
Optimization (continued)

- Notice that constraints are equivalent to \( \xi_i = \max(0, 1 - l_i(w^T x_i + b)) \)
- set \( x_i = [x_i; 1] \) and \( w = [w; b] \) to simplify.

### Reformulation

\[
\text{Minimize}_w J(w) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \max(0, 1 - l_i(w^T x_i))
\]

- Unconstrained optimization
Optimization (continued)

- Notice that constraints are equivalent to \( \xi_i = \max(0, 1 - l_i(w^T x_i + b)) \)
- set \( x_i = [x_i; 1] \) and \( w = [w; b] \) to simplify.

Reformulation

\[
\text{Minimize}_w J(w) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \max(0, 1 - l_i(w^T x_i))
\]

- Unconstrained optimization
- Gradient descent \( w_{t+1} = w_t - \nu_t \nabla_w J(w_t) \)
Stochastic gradient descent

A gradient computed per sample:

\[
J(w, x_i, l_i) = \frac{1}{2} w^T w + C \max(0, 1 - l_i(w^T x_i))
\]

- initialization \( w_0 = 0 \)
- While not converged
  - For each training sample \((x_i, l_i)\)
  - Compute \( \nabla J(w_t, x_i, l_i) \)
  - Update \( w_{t+1} = w_t - \nu_t \nabla w \)
- Return \( w \)
Problem

\( J \) is not differentiable!

\[
\nabla J = \begin{cases} 
  w & \text{if max}(0, 1 - l_i w^T x_i) = 0 \\
  w - Cl_i x_i & \text{otherwise}
\end{cases}
\]

\[
\text{initialization} \quad w_0 = 0
\]

While not converged

/\begin{align*}
\text{For each training sample} \quad (x_i, l_i) \\
\text{If} \quad l_i w^T x_i \leq 1, \quad w_{t+1} = (1 - \nu_t) w_t + \nu_t Cl_i x_i \\
\text{Otherwise} \quad w_{t+1} = (1 - \nu_t) w_t
\end{align*}/

\]

\text{Return} \quad w

Stochastic Gradient Descent

Show the training set before picking an example
Problem

\( J \) is not differentiable! Strategy:

- \( \nabla J = w \) if \( \max(0, 1 - l_i w^T x_i) = 0 \)
- \( \nabla J = w - C l_i x_i \) otherwise
Optimization (continued)

Problem

\( J \) is not differentiable! Strategy:

- \( \nabla J = \mathbf{w} \) if \( \max(0, 1 - l_i \mathbf{w}^T \mathbf{x}_i) = 0 \)
- \( \nabla J = \mathbf{w} - C l_i \mathbf{x}_i \) otherwise

- Initialization \( \mathbf{w}_0 = 0 \)
- While not converged
  - For each training sample \((\mathbf{x}_i, l_i)\)
    - If \( l_i \mathbf{w}_t^T \mathbf{x}_i \leq 1 \), \( \mathbf{w}_{t+1} = (1 - \nu_t) \mathbf{w}_t + \nu_t C l_i \mathbf{x}_i \)
    - Otherwise \( \mathbf{w}_{t+1} = (1 - \nu_t) \mathbf{w}_t \)
- Return \( \mathbf{w} \)
Optimization (continued)

Problem

\[ J \text{ is not differentiable! Strategy:} \]

- \[ \nabla J = w \text{ if } \max(0, 1 - l_i w^T x_i) = 0 \]
- \[ \nabla J = w - Cl_i x_i \text{ otherwise} \]

- initialization \( w_0 = 0 \)
- While not converged
  - For each training sample \((x_i, l_i)\)
  - If \( l_i w^T x_i \leq 1 \), \( w_{t+1} = (1 - \nu_t) w_t + \nu_t C l_i x_i \)
  - Otherwise \( w_{t+1} = (1 - \nu_t) w_t \)
- Return \( w \)

Stochastic Gradient Descent

Shuffle the training set before picking an example
Pedestrian detection (Dalal & Triggs 2005)

- Descriptor of each image: Histogram of oriented gradients
- Classified using a linear SVM (soft: allows for some margin violation during training).
Decision trees

**Principle**
Partition the feature space into a set of regions on which the decision will be uniform.

- How to build a decision tree:
  - Given a set of training data with associated labels \((x_i, l_i)_i\)
Decision trees

Principle

Partition the feature space into a set of regions on which the decision will be uniform.

- How to build a decision tree:
  - Given a set of training data with associated labels \((x_i, l_i)_i\)
  - At each step, find the subdivision of the space that best enforces the homogeneity of the labels inside each domain using a (usually weak) classifier.
Decision trees

Principle
Partition the feature space into a set of regions on which the decision will be uniform.

How to build a decision tree:
- Given a set of training data with associated labels \((x_i, l_i)_i\)
- At each step, find the subdivision of the space that best enforces the homogeneity of the labels inside each domain using a (usually weak) classifier.

Homogeneity measure
Example: Shannon Entropy. Let random variable of the label of the samples inside a domain \(\Omega\):

\[
H(X) = - \sum_{l} P(X = l) \log P(X = l)
\]

The entropy should be as low as possible inside each partition cell.
Random Forests [Breiman, Cutler 2001]

<table>
<thead>
<tr>
<th>Principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate the result of several decision trees.</td>
</tr>
</tbody>
</table>
Random Forests [Breiman, Cutler 2001]

**Principle**

Aggregate the result of several decision trees.

- Advantage: less overfitting.
Random Forests [Breiman, Cutler 2001]

**Principle**
Aggregate the result of several decision trees.

- Advantage: less overfitting.

**Training Algorithm**
- Create $B$ training sets by drawing $N$ samples from the initial training set.
Random Forests [Breiman, Cutler 2001]

**Principle**
Aggregate the result of several decision trees.

- Advantage: less overfitting.

**Training Algorithm**
- Create B training sets by drawing N samples from the initial training set.
- On each of the B training sets train a decision tree.
**Random Forests [Breiman, Cutler 2001]**

**Principle**

Aggregate the result of several decision trees.

- Advantage: less overfitting.

**Training Algorithm**

- Create B training sets by drawing N samples from the initial training set.
- On each of the B training sets train a decision tree.

**Decision Algorithm**

- Apply the B trees to a data to classify and record the B decisions.
Random Forests [Breiman, Cutler 2001]

**Principle**
Aggregate the result of several decision trees.

- Advantage: less overfitting.

**Training Algorithm**
- Create B training sets by drawing N samples from the initial training set.
- On each of the B training sets train a decision tree.

**Decision Algorithm**
- Apply the B trees to a data to classify and record the B decisions
- Majority vote for the final decision.
Real-Time Human Pose Recognition in Parts from Single Depth Images

Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, Andrew Blake

CVPR 2011
Body part recognition

right hand

neck

left shoulder

right elbow
Body part recognition

- No temporal information
  - frame-by-frame

- Local pose estimate of parts
  - each pixel & each body joint treated independently

- Very fast
  - simple depth image features
  - parallel decision forest classifier
The Kinect pose estimation pipeline

capture depth image & remove bg

infer body parts per pixel

cluster pixels to hypothesize body joint positions

fit model & track skeleton
Synthetic training data

Record mocap
500k frames distillé to 100k poses

Retarget to several models

Render (depth, body parts) pairs

Train invariance to:
Synthetic vs. real data

synthetic
(train & test)

real
(test)
Fast depth image features

- Depth comparisons
  - very fast to compute

\[ f(I, x) = d_I(x) - d_I(x + \Delta) \]
\[ \Delta = v/d_I(x) \]

feature response
image coordinate

scales inversely with depth

Background pixels
\( d = \text{large constant} \)
Depth of trees

input depth

ground truth parts

inferred parts (soft)

depth 18
Number of trees

Average per-class

40% 45% 50% 55%

Number of trees

1 2 3 4 5 6

ground truth

inferred body parts (most likely)
1 tree 3 trees 6 trees
input depth

inferred body parts

front view

side view

inferred joint positions

top view

no tracking or smoothing
input depth
inferred body parts

front view
side view
top view

inferred joint positions

no tracking or smoothing
From proposals to skeleton

- Use...
  - 3D joint hypotheses
  - kinematic constraints
  - temporal coherence

- ... to give
  - full skeleton
  - higher accuracy
  - invisible joints
  - multi-player
Boosting

**Principle [Rojas 2009]**

Combine $N$ weak classifiers $f_n$ to get a better classifier $g = \sum_{i=1}^{N} \alpha_i f_i$. Initially all training samples have the same weight. At each iteration:

1. Find the classifier with the lowest training error.
2. Raise the weights of the training examples misclassified by this classifier.

Final classifier: linear combination of the classifiers (weights proportional to the classifier accuracy). Examples: AdaBoost, LogitBoost; the difference lies in the way the weights of the samples are updated.
Boosting

Principle [Rojas 2009]

Combine $N$ weak classifiers $f_n$ to get a better classifier $g = \sum_{i=1}^{N} \alpha_i f_i$.

- Initially all training samples have the same weight.
Principle [Rojas 2009]

Combine $N$ weak classifiers $f_n$ to get a better classifier $g = \sum_{i=1}^{N} \alpha_i f_i$.

- Initially all training samples have the same weight.
- At each iteration:
  - Find the classifier with the lowest training error.
Boosting

**Principle [Rojas 2009]**

Combine $N$ weak classifiers $f_n$ to get a better classifier $g = \sum_{i=1}^{N} \alpha_i f_i$.

- Initially all training samples have the same weight.
- At each iteration:
  - Find the classifier with the lowest training error.
  - Raise the weights of the training examples misclassified by this classifier.
Boosting

**Principle [Rojas 2009]**

Combine \( N \) weak classifiers \( f_i \) to get a better classifier \( g = \sum_{i=1}^{N} \alpha_i f_i \).

- Initially all training samples have the same weight.
- At each iteration:
  - Find the classifier with the lowest training error.
  - Raise the weights of the training examples misclassified by this classifier.
- Final classifier: linear combination of the classifiers (weights proportional to the classifier accuracy).
Boosting

Principle [Rojas 2009]

Combine \( N \) \emph{weak} classifiers \( f_n \) to get a better classifier \( g = \sum_{i=1}^{N} \alpha_i f_i \).

- Initially all training samples have the same weight.
- At each iteration:
  - Find the classifier with the lowest training error.
  - Raise the weights of the training examples misclassified by this classifier.
- Final classifier: linear combination of the classifiers (weights proportional to the classifier accuracy).

Examples

Adaboost, Logiboost: difference lies in the way the weights of the samples are updated.
Example

Image David Filliat (ENSTA)
Example

Weights Increased

Image David Filliat (ENSTA)
Example

Image David Filliat (ENSTA)
Example

Image David Filliat (ENSTA)
Example

Final classifier is a combination of weak classifiers

Image David Filliat (ENSTA)
Outline

1. Introduction
2. General Formulation
3. Support Vector Machine
4. On Trees and Forests
5. Boosting
7. Neural nets
8. Generative problems
9. Generative Adversarial Networks (GAN)
Sparse processing of signals

Signal Processing aims to decompose complex signals using elementary functions which are then easier to manipulate.

\[ x(t) = \sum_{i=-\infty}^{\infty} \alpha_i \varphi_i(t) \]
Between two representations of a signal pick the ones with the higher number of zero coefficients.
Between two representations of a signal pick the ones with the higher number of zero coefficients.
Patch-based approaches for images and surfaces

- Texture synthesis [Efros 99], Non local means [Buades et al. 2005].

**Compressive sensing theory [Candès et al. 2006]**

There exists spaces, in which the signals would be sparsely represented, that are especially well suited for processing the signals.

- Sparse regularization for image analysis, inpainting... [Elad et al. 2006], [Mairal 2009] The K-SVD algorithm
**Sparse Coding: A brief reminder on norms**

**Norm definition**

Let $E$ be a vector space over a subfield $K$, a norm on $E$ is an application with nonnegative values $\| \cdot \| : E \to \mathbb{R}$ such that for all $\alpha \in K$ and $u, v \in E$:

- $\| \alpha v \| = |\alpha| \| v \|$ (positive homogeneity)
- $\| u + v \| \leq \| u \| + \| v \|$ (subadditivity)
- $\| u \| = 0_K \iff u = 0_E$ (separation)

The $\ell_2$ norm is also called the euclidean norm. Let $x$ be a vector in $\mathbb{R}^n$ with coordinates $(x_1, \ldots, x_n)$ in the canonical basis, the $\ell_2$ norm writes:

$$\| x \|_2 = \sqrt{x \cdot x^T} = \left( \sum_{i=1}^{n} x_i^2 \right)^{1/2}$$
Norm definition

Let $E$ be a vector space over a subfield $K$, a norm on $E$ is an application with nonnegative values $\| \cdot \| : E \to R$ such that for all $\alpha \in K$ and $u, v \in E$:

- $\| \alpha v \| = |\alpha| \| v \|$ (positive homogeneity)
- $\| u + v \| \leq \| u \| + \| v \|$ (subadditivity)
- $\| u \| = 0_K \iff u = 0_E$ (separation)

The $\ell^2$ norm is also called the euclidean norm. Let $x$ be a vector in $\mathbb{R}^n$ with coordinates $(x_1, \cdots, x_n)$ in the canonical basis, the $\ell^2$ norm writes:

$$\| x \|_2 = \sqrt{x \cdot x^T} = \left( \sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$$
Norm Examples on vectors of \( \mathbb{R}^n \)

- \( \ell^1 \) Norm (Manhattan)

\[
\|x\|_1 = \left( \sum_{i=1}^{n} |x_i| \right)
\]
Norm Examples on vectors of $\mathbb{R}^n$

- $\ell^1$ Norm (Manhattan)
  $$\|x\|_1 = \left(\sum_{i=1}^{n} |x_i|\right)$$

- $\ell^3$
  $$\|x\|_3 = \left(\sum_{i=1}^{n} |x_i|^3\right)^{\frac{1}{3}}$$

Exercice: Prove that $\ell^\infty$ is indeed a norm?
Norm Examples on vectors of $\mathbb{R}^n$

- $\ell^1$ Norm (Manhattan)
  \[ \| x \|_1 = \left( \sum_{i=1}^{n} |x_i| \right) \]

- $\ell^3$
  \[ \| x \|_3 = \left( \sum_{i=1}^{n} |x_i|^3 \right)^{\frac{1}{3}} \]

- $\ell^{2.1}$:
  \[ \| x \|_{2.1} = \left( \sum_{i=1}^{n} x_i^{2.1} \right)^{\frac{1}{2.1}} \]
Norm Examples on vectors of $\mathbb{R}^n$

- $\ell^1$ Norm (Manhattan)
  \[
  \|x\|_1 = \left( \sum_{i=1}^{n} |x_i| \right)
  \]

- $\ell^3$
  \[
  \|x\|_3 = \left( \sum_{i=1}^{n} |x_i|^3 \right)^{\frac{1}{3}}
  \]

- $\ell^{2.1}$
  \[
  \|x\|_{2.1} = \left( \sum_{i=1}^{n} x_i^{2.1} \right)^{\frac{1}{2.1}}
  \]

- $\ell^p$ pour $p \geq 1$
  \[
  \|x\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}
  \]

Exercice: Prove that $\ell^\infty$ is indeed a norm?
Norm Examples on vectors of $\mathbb{R}^n$

- $\ell^1$ Norm (Manhattan)

  $$\|x\|_1 = \left(\sum_{i=1}^{n} |x_i|\right)$$

- $\ell^3$

  $$\|x\|_3 = \left(\sum_{i=1}^{n} |x_i|^3\right)^{\frac{1}{3}}$$

- $\ell^{2.1}$

  $$\|x\|_{2.1} = \left(\sum_{i=1}^{n} x_i^{2.1}\right)^{\frac{1}{2.1}}$$

- $\ell^p$ pour $p \geq 1$

  $$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$

- $\ell^\infty$

  $$\|x\|_\infty = \max_{i=1\ldots n} |x_i|$$

Exercice: Prove that $\ell^\infty$ is indeed a norm?
The ball of radius 1 for norms $\ell^p$ with $p \geq 2$
The ball of radius 1 for norms $\ell^p$ with $p \geq 2$
The ball of radius 1 for norms $\ell^p$ with $p \geq 2$
The ball of radius 1 for norms $\ell^p$ with $p \geq 2$
The ball of radius 1 for norms $\ell^p$ with $p \leq 2$
The ball of radius 1 for norms $\ell^p$ with $p \leq 2$
The ball of radius 1 for norms $\ell^p$ with $p \leq 2$
The ball of radius 1 for norms $\ell^p$ with $p \leq 2$
The ball of radius 1 with norms and quasi-norms $\ell^p$
Sparsity definition

A vector $x \in \mathbb{R}^N$ is said to be $s$-sparse if at most $s$ of its entries are non zero, i.e.

$$\text{card} \ support(x) \leq s$$

where $support(x) = \{i | x_i \neq 0\}$. We note $\|x\|_0 = \text{card} \ support(x)$ and call it $\ell^0$. 
Norm and sparsity

**Sparsity definition**

A vector \( x \in \mathbb{R}^N \) is said to be \( s \)-sparse if at most \( s \) of its entries are non zero, i.e.

\[
\text{card support}(x) \leq s
\]

where \( \text{support}(x) = \{i | x_i \neq 0\} \).

We note \( \|x\|_0 = \text{card support}(x) \) and call it \( \ell^0 \).

- Is \( \ell^0 \) a norm?
Sparse Coding with the $\ell^0$ norm

### Problem statement

Let $A \in \mathbb{R}^{m \times n}$ and $\alpha$ a $s$-sparse vector in $\mathbb{R}^n$. Let $x \in \mathbb{R}^m$ such that $x = A\alpha$. Assume only $x$ and $A$ are known and we want to recover $\alpha$. If $m < n$, the system is underdetermined.
Sparse Coding with the $\ell^0$ norm

Problem statement

Let $A \in \mathbb{R}^{m \times n}$ and $\alpha$ a $s$-sparse vector in $\mathbb{R}^n$. Let $x \in \mathbb{R}^m$ such that $x = A\alpha$. Assume only $x$ and $A$ are known and we want to recover $\alpha$. If $m < n$, the system is underdetermined.

Sparsity hypothesis

Identifying the solution $\alpha$ under the $s$-sparsity hypothesis is easier.
Sparse Coding with the $\ell^0$ norm

**Optimization problem**

Given a measurement matrix $A \in \mathbb{R}^{m \times n}$ and $x$ a vector in $\mathbb{R}^n$, under the $s$-sparse assumption, the vector $\alpha$ can be reconstructed as the solution of:

$$\text{Minimize} \| \alpha \|_0$$

$$\text{s.t. } x = A\alpha$$

$(P_0)$
Sparse Coding with the $\ell^0$ norm

**Optimization problem**

Given a measurement matrix $A \in \mathbb{R}^{m \times n}$ and $x$ a vector in $\mathbb{R}^n$, under the $s$-sparse assumption, the vector $\alpha$ can be reconstructed as the solution of:

$$\begin{align*}
\text{Minimize} & \quad \|\alpha\|_0 \\
\text{s.t.} & \quad x = A\alpha
\end{align*}$$

$(P_0)$

- Nonconvex optimization problem
Problem \((P_0)\) is a NP-hard problem

- Reformulate the problem as

\[
\begin{align*}
\text{Minimize} & \quad \|\alpha\|_0 \\
\text{s.t.} & \quad \|x - A\alpha\|_2 \leq \eta
\end{align*}
\]

\((P_{0,\eta})\)

**Theorem**

Problem \((P_{0,\eta})\) is a NP-hard problem

- NP-hardness: all problems for which a solving algorithm could be turned in polynomial time into a solving algorithm for any NP-problem.
- Proof: demonstrate that using Problem \((P_{0,\eta})\) one can solve for the exact cover 3-set problem.
- Reminder: Given a collection \(S\) of 3-subsets of a set \(X\), an exact cover of \(X\) is a subcollection \(S_{sub}\) of \(S\) such that the intersection of two distinct elements of \(S_{sub}\) is empty and the union of all elements of \(S_{sub}\) cover \(X\).
Sparse decomposition algorithm

Sparse Decomposition

Given a dictionary $D \in \mathbb{R}^{m,n}$ whose columns have norm 1 and a signal $x \in \mathbb{R}^n$ find a vector $\alpha$ whose sparsity is $s$ minimizing $\|x - D\alpha\|_2^2$

- Efficient greedy algorithms have been proposed to find an approximate solution.
Matching Pursuit

Matching Pursuit Algorithm [Mallat & Zhang 1993]

- Set $k = 0$, $\alpha = 0_{\mathbb{R}^n}$
- While $k < s$ and $\|x - D\alpha\| > 0$ do:
  - Select index $j$ maximizing $|D_j^T \cdot (x - D\alpha)|$
  - Update coefficients $\alpha(j) = \alpha(j) + D_j^T \cdot (x - D\alpha)$

At each step the algorithm finds the atom that best represents the residual $r = x - D\alpha$. The residual monotonically decreases until the residual is orthogonal to all $D_j$. How does the sparsity behave? Beyond classication: Dictionary Learning
Matching Pursuit

Matching Pursuit Algorithm [Mallat & Zhang 1993]

- Set $k = 0$, $\alpha = 0_{\mathbb{R}^n}$
- While $k < s$ and $\|x - D\alpha\| > 0$ do:
  - Select index $j$ maximizing $|D_j^T \cdot (x - D\alpha)|$
  - Update coefficients $\alpha(j) = \alpha(j) + D_j^T \cdot (x - D\alpha)$

- At each step the algorithm finds the atom that best represents the residual $r = x - D\alpha$
Matching Pursuit

Matching Pursuit Algorithm [Mallat & Zhang 1993]

- Set $k = 0, \alpha = 0_{\mathbb{R}^n}$
- While $k < s$ and $\|x - D\alpha\| > 0$ do:
  - Select index $j$ maximizing $|D_j^T \cdot (x - D\alpha)|$
  - Update coefficients $\alpha(j) = \alpha(j) + D_j^T \cdot (x - D\alpha)$

- At each step the algorithm finds the atom that best represents the residual $r = x - D\alpha$
- The residual monotonically decreases until the residual is orthogonal to all $D_j$
Matching Pursuit

Matching Pursuit Algorithm [Mallat & Zhang 1993]

- Set $k = 0$, $\alpha = 0_{\mathbb{R}^n}$
- While $k < s$ and $\|x - D\alpha\| > 0$ do:
  - Select index $j$ maximizing $|D_j^T \cdot (x - D\alpha)|$
  - Update coefficients $\alpha(j) = \alpha(j) + D_j^T \cdot (x - D\alpha)$

- At each step the algorithm finds the atom that best represents the residual $r = x - D\alpha$
- The residual monotonically decreases until the residual is orthogonal to all $D_j$

How does the sparsity behave?
Matching Pursuit

Matching Pursuit Algorithm [Mallat & Zhang 1993]

- Set $k = 0$, $\alpha = 0_{\mathbb{R}^n}$
- While $k < s$ and $\|x - D\alpha\| > 0$ do:
  - Select index $j$ maximizing $|D_j^T \cdot (x - D\alpha)|$
  - Update coefficients $\alpha(j) = \alpha(j) + D_j^T \cdot (x - D\alpha)$

- At each step the algorithm finds the atom that best represents the residual $r = x - D\alpha$
- The residual monotonically decreases until the residual is orthogonal to all $D_j$

How does the sparsity behave? nondecreasing
Orthogonal Matching Pursuit

- Goal: The sparsity should increase at each step.
Orthogonal Matching Pursuit

- **Goal:** The sparsity should increase at each step.

- **How?**

```
Orthogonal Matching Pursuit

k = 0, α = 0
R, Γ = ∅

While k < s and x − Dα > 0 do:
  Select index j maximizing D^T j ⋅ (x − Dα)
  Update the active set Γ = Γ ∪ {j}
  Recompute α ∈ Γ minimizing x − DΓα
  Set α̅Γ = 0

Remark: DΓ, αΓ: matrix (resp. vector) composed of the columns (resp. elements) of D (resp. α) whose indices are in Γ.
```
Orthogonal Matching Pursuit

- **Goal:** The sparsity should increase at each step.
- **How?** Render the residual orthogonal to all selected atoms.
Orthogonal Matching Pursuit

- **Goal:** The sparsity should increase at each step.

- **How?** Render the residual orthogonal to all selected atoms.
Orthogonal Matching Pursuit

- **Goal:** The sparsity should increase at each step.
- **How?** Render the residual orthogonal to all selected atoms.

Orthogonal Matching Pursuit (OMP)

- Set $k = 0$, $\alpha = 0_{\mathbb{R}^n}$, $\Gamma = \emptyset$
- While $k < s$ and $\|x - D\alpha\| > 0$ do:
  - Select index $j$ maximizing $|D_j^T \cdot (x - D\alpha)|$
  - Update the active set $\Gamma = \Gamma \cup \{j\}$
  - Recompute $\alpha_\Gamma$ minimizing $x - D_\Gamma \alpha_\Gamma$
  - Set $\alpha_{\bar{\Gamma}} = 0$

**Remark:** $D_\Gamma$, $\alpha_\Gamma$: matrix (resp. vector) composed of the columns (resp. elements) of $D$ (resp. $\alpha$) whose indices are in $\Gamma$. 
What can we prove about OMP?

- The index selection is guided by finding the one that makes the error decrease most.
What can we prove about OMP?

- The index selection is guided by finding the one that makes the error decrease most.
- What about the case where a vector is a linear combination of 3 vectors.
What can we prove about OMP?

- The index selection is guided by finding the one that makes the error decrease most.
- What about the case where a vector is a linear combination of 3 vectors.
- Tropp and Gilbert (2007): OMP is able to reliably recover a sparse vector from random measurements.
What can we prove about OMP?

- The index selection is guided by finding the one that makes the error decrease most.
- What about the case where a vector is a linear combination of 3 vectors.
- Tropp and Gilbert (2007): OMP is able to reliably recover a sparse vector from random measurements.
- OMP is slower than MP
How can we make OMP faster?

Which step is computationally intensive?

- Computing the best index means computing $D_f^T (x - D_f \alpha_f)$ and taking the index of the smallest coefficient.
How can we make OMP faster?

Which step is computationally intensive?

- Computing the best index means computing $D_T^T(x - D_\Gamma \alpha_\Gamma)$ and taking the index of the smallest coefficient.
- Then we compute $\alpha_{\Gamma'}$ as $\alpha$ minimizing $\|x - D_{\Gamma'} \alpha\|^2$.
How can we make OMP faster?

Which step is computationally intensive?

- Computing the best index means computing $D_f^T(x - D_\Gamma \alpha_\Gamma)$ and taking the index of the smallest coefficient.
- Then we compute $\alpha_{\Gamma'}$ as $\alpha$ minimizing $\|x - D_{\Gamma'} \alpha\|^2$
- Closed form solution:
  \[
  \alpha_{\Gamma'} = (D_{\Gamma'}^T D_{\Gamma'})^{-1} D_{\Gamma'} x
  \]
How can we make OMP faster?

Which step is computationally intensive?

- Computing the best index means computing $D_f^T (x - D_G \alpha_G)$ and taking the index of the smallest coefficient.
- Then we compute $\alpha_{G'}$ as $\alpha$ minimizing $\|x - D_{G'} \alpha\|^2$.
- Closed form solution:
  $$\alpha_{G'} = (D_{G'}^T D_{G'})^{-1} D_{G'} x$$

Making OMP faster

Invert quickly $D_f^T D_f$, knowing the inverse of $D_f^T D_f$. 
Update of the inverse of $D^T D$ when appending a column $d$

- $u_1 \leftarrow D^T d$
- $u_2 \leftarrow (D^T D)^{-1} u_1$
- $u_3 \leftarrow d u_2$
- $A \leftarrow (X^T X)^{-1} + d u_2^T u_2$
- $s \leftarrow \frac{1}{d^T d - u_1^T u_2}$
- Updated inverse: $(A - u_3 - u_3^T s)$
An application of OMP: synthesizing terrains based on examples [Guérin et al. 2016]

A terrain is seen as a set of blended patches

Terrain model $\mathcal{T}$

Sparse Primitives

$\Omega_i$ 

$\Omega_j$

$f_i(p)$

$f_j(p)$
An application of OMP: synthesizing terrains based on examples [Guérin et al. 2016]

- Build a dictionary by decomposing a real-world elevation map into patches
- Decompose patches to synthesize on it
Terrain “Amplification”

[Diagram showing high resolution input terrain, dictionary creation, down sampling to high resolution H, and further down sampling to low resolution L]

Remarks: It works because terrains are height fields, much more complicated with images or textures!
Terrain “Amplification”

**Remark**

Works because terrains are heightfields, much more complicated with images or textures!
Dictionary learning

Problem

In the context of Image Processing and Synthesis, we only have access to a set of signals for which we want to build a dictionary.
Dictionary learning

Problem

In the context of Image Processing and Synthesis, we only have access to a set of signals for which \textbf{we want to build a dictionary.}

Dictionary Learning Problem

Given a set of signals $x_i$ for $i = 1 \cdots N$ in $\mathbb{R}^n$ we want to build a matrix $D \in \mathbb{R}^{n \times m}$ and coefficients $\alpha_i \in \mathbb{R}^m$ for $i = 1 \cdots n$ solving:

Minimize $\sum_{i=1}^{N} \| x_i - D \cdot \alpha_i \|_2^2$ \hspace{1cm} \text{(P$D$,$\alpha$,$0$)}

\[\begin{align*}
D \in \mathbb{R}^{n \times m} \\
\|D_i\|_2 \leq 1 \\
\forall i = 1 \cdots N, \alpha_i \in \mathbb{R}^m, \\
\forall i = 1 \cdots N, \|\alpha_i\|_0 = s
\end{align*}\]
Dictionary learning problems

- Still a nonconvex problem
- Common approach: alternate minimization
  - Fix the dictionary $D$ and compute the sparse decomposition $\alpha$
  - Fix the sparse decomposition $\alpha$ and compute $D$
Method of Optimal Directions (MOD)

- First introduced by Engan et al. [1999]
Method of Optimal Directions (MOD)

- First introduced by Engan et al. [1999]
- Step 1: Compute the sparse codes?
Method of Optimal Directions (MOD)

- First introduced by Engan et al. [1999]
- Step 1: Compute the sparse codes? MP, OMP, Iterative Hard Thresholding
Method of Optimal Directions (MOD)

- First introduced by Engan et al. [1999]
- Step 1: Compute the sparse codes? MP, OMP, Iterative Hard Thresholding
- Step 2: Update the dictionary
MOD: Dictionary Update

Assume all coefficients $\alpha_i$ are fixed, Problem $(P_{D,\alpha,0})$ becomes

$$\text{Minimize} \sum_{i=1}^{N} \| x_i - D \cdot \alpha_i \|_2^2$$

This problem is a convex problem on a convex set. Discarding the 1-norm constraint yields a least squares objective. Idea: Solve the least squares problem and project the solution onto the convex set of admissible solutions.
MOD: Dictionary Update

- Assume all coefficients $\alpha_i$ are fixed, Problem $(P_{D,\alpha,0})$ becomes

  $$\text{Minimize} \sum_{D \in \mathbb{R}^{n \times m}, \|D_i\|_2 \leq 1}^N \| x_i - D \cdot \alpha_i \|_2^2$$

- This problem is a convex problem on a convex set
MOD: Dictionary Update

- Assume all coefficients $\alpha_i$ are fixed, Problem $(P_{D,\alpha,0})$ becomes

$$\text{Minimize } \sum_{i=1}^{N} \| x_i - D \cdot \alpha_i \|_2^2$$

- This problem is a convex problem on a convex set
- Discarding the 1-norm constraint yields a least squares objective
MOD: Dictionary Update

- Assume all coefficients $\alpha_i$ are fixed, Problem $(P_{D,\alpha,0})$ becomes

$$\text{Minimize} \sum_{i=1}^{N} \| x_i - D \cdot \alpha_i \|_2^2$$

- This problem is a convex problem on a convex set

- Discarding the 1-norm constraint yields a least squares objective

- Idea: Solve the least squares problem and project the solution onto the convex set of admissible solutions
Assume all coefficients $\alpha_i$ are fixed, Problem $(P_{D,\alpha,0})$ becomes

$$\text{Minimize} \sum_{i=1}^{N} \| x_i - D \cdot \alpha_i \|_2^2$$

This problem is a convex problem on a convex set.

Discarding the 1-norm constraint yields a least squares objective.

Idea: Solve the least squares problem and project the solution onto the convex set of admissible solutions.

Bertsekas 1999

In general, solving the general problem and projecting the solution on the convex constraints set yields a poor solution.
Assume all coefficients $\alpha_i$ are fixed, Problem $(P_{D,\alpha,0})$ becomes

$$
\text{Minimize} \sum_{i=1}^{N} \left\| x_i - D \cdot \alpha_i \right\|^2_2
\text{subject to} \| D_i \|_2 \leq 1
$$

This problem is a convex problem on a convex set.

Discarding the 1-norm constraint yields a least squares objective.

Idea: Solve the least squares problem and project the solution onto the convex set of admissible solutions.

---

Bertsekas 1999

Since the $\ell^0$ norm remains constant when a vector undergoes a nonzero rescaling, the projection is valid.
Dictionary Update

Least Squares Problem

Solve for $D$ in:

\[
\text{Minimize} \sum_{i=1}^{N} \|x_i - D\alpha_i\|^2_2
\]
Dictionary Update

Least Squares Problem

Solve for $D$ in:

$$\text{Minimize} \sum_{i=1}^{N} \|x_i - D\alpha_i\|_2^2$$

- Compute the gradient and set to 0:
  $$\sum_{i=1}^{N} (x_i - D\alpha_i)\alpha_i^T = 0$$

Projection on the constraint set: normalizing each column of $D$ if its norm is above 1.
Least Squares Problem

Solve for $D$ in:

$$
\text{Minimize}_{D \in \mathbb{R}^{m \times n}} \sum_{i=1}^{N} \left\| x_i - D \alpha_i \right\|_2^2
$$

- Compute the gradient and set to 0: $\sum_{i=1}^{N} (x_i - D \alpha_i) \alpha_i^T = 0$

- $D = (\sum_{i=1}^{N} x_i \alpha_i^T) (\sum_{i=1}^{N} (\alpha_i \alpha_i^T)^{-1})^{-1}$
Least Squares Problem

Solve for $D$ in:

$$\text{Minimize} \sum_{i=1}^{N} \| x_i - D\alpha_i \|_2^2$$

- Compute the gradient and set to 0:
  $$\sum_{i=1}^{N} (x_i - D\alpha_i)\alpha_i^T = 0$$

- $D = (\sum_{i=1}^{N} x_i\alpha_i^T)(\sum_{i=1}^{N} (\alpha_i\alpha_i^T)^{-1}$

- Setting $A = (\alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_N)$, $X = (x_1 \mid x_2 \mid \cdots \mid x_N)$ one has: $D = XA^T(AA^T)^{-1}$
**Least Squares Problem**

Solve for $D$ in:

$$\text{Minimize} \sum_{i=1}^{N} \|x_i - D\alpha_i\|_2^2$$

- Compute the gradient and set to 0: $$\sum_{i=1}^{N} (x_i - D\alpha_i)\alpha_i^T = 0$$
- $$D = (\sum_{i=1}^{N} x_i\alpha_i^T)(\sum_{i=1}^{N}(\alpha_i\alpha_i^T)^{-1}$$
- Setting $A = (\alpha_1 | \alpha_2 | \cdots | \alpha_N), X = (x_1 | x_2 | \cdots | x_N)$, one has: $D = XA^T(AA^T)^{-1}$
- Projection on the constraint set: normalizing each column of $D$ if its norm is above 1.
K-SVD algorithm

- Still an alternating direction minimization method
K-SVD algorithm

- Still an alternating direction minimization method
- Goal: Incorporate the sparsity constraint also in the dictionary update step
Goal

- A set of training signals \( \{x_i\}_{i=1}^{N} \in \mathbb{R}^n \)
- Design a dictionary \( D \in \mathbb{R}^{n \times K} \) such that there exists \( \alpha \in \mathbb{R}^k \) such that either \( x = D\alpha \) or \( \|x - D\alpha\|_p \leq \varepsilon \)
- if \( n < K \) and \( D \) is full-ranked the solution must be constrained
  - \( \min_\alpha \|\alpha\|_0 \) s.t. \( x = D\alpha \)
  - \( \min_\alpha \|\alpha\|_0 \) s.t. \( \|x - D\alpha\|_2 \leq \varepsilon \)
- Design \( D \) in order to best fit the sparsity model imposed
An extension of K-means

- K-means search for the best possible representative enforcing that each representation uses a single atom with coefficient 1.
- K-SVD solves \[ \min_{D,A} \|X - DA\|_F^2 \text{ s.t. } \forall i \|\alpha_i\|_0 \leq T_0 \]
- An iterative approach that alternates between two steps
  - Sparse coding of the examples based on the current dictionary
  - Update of the dictionary so as to better fit the data
- \( X \in \mathbb{R}^{n \times N} \): training samples, \( A \in \mathbb{R}^{K \times N} \) matrix of coefficients
Sparse Coding stage

- $D$ is fixed, compute the best representation $\alpha_i$ of sample $x_i$
- Find $\alpha_i$ minimizing $\| x_i - D\alpha_i \|_2^2$ s.t. $\| \alpha_i \|_0 \leq T_0$
- Can be done using a pursuit algorithm (e.g. Orthogonal Matching Pursuit)
Dictionary Update stage

- The update will be done atom by atom.
- \[ \| X - DA \|_F^2 = \| X - \sum_{j=1}^{N} d_j \alpha_j^T \|_F^2 = \| X - \sum_{j=1, j \neq k}^{N} d_j \alpha_j^T - d_k \alpha_k^T \|_F^2 \]
- \[ E_k = X - \sum_{j=1, j \neq k}^{N} d_j \alpha_j^T \] error obtained by omitting atom \( d_k \) in the decomposition
- Finally solve for:
  \[ \| E_k - d_k \alpha_k^T \|_F \] w.r.t. \( d_k, \alpha_k^T \)
- Solve using SVD? if so sparsity not enforced.
Trick to enforce the sparsity

\[ \omega_k = \{i | 1 \leq i \leq K, \alpha_T^k(i) \neq 0\} \]

- Restrict \( E_k \) and \( \alpha_T^k \) to \( E_R^k \) and \( \alpha_R^k \) by selecting only the columns of indices included in the support of \( \alpha_T^k \).

- Use SVD to decompose \( E_R^k = U \Delta V^T \).

- Set \( d_k \) to be the first column of \( U \).

- Set \( \alpha_R^k \) to be the first column of \( V \) multiplied by \( \Delta(1,1) \).

- The columns of \( D \) remain normalized and the support of the representations can not increase.
Application to the denoising of images

- noisy input image $x$
- Build $\hat{D}$ and $(\hat{\alpha}_i)_i$ the dictionary and representations of all patches of image $x$
- $(P_i(y))_i$ the set of all image $y$ patches.
- $\hat{D}\hat{\alpha}_i$ is the representation of patch $P_i(x)$
- Find $y$ minimizing
  \[
  \lambda \| x - y \|_2^2 + \sum_i \| \hat{D}\hat{\alpha}_i - P_i(y) \|_2^2
  \]
  fidelity term
  proximity of the reconstruction to the denoised patch
Can be tested on IPOL http://www.ipol.im/pub/algo/llm_ksvd
Learned dictionary

Dictionary learned from face patches
Train time 9.0s on 94500 patches
Learned Color dictionary
Denoising via dictionary learning

noisy image
Denoising via dictionary learning

Denoising each channel separately
Denoising via dictionary learning
Denoising via dictionary learning

Denoising each channel separately (left) vs globally (right)
Comparison to NL-means

Original
Comparison to NL-means
Comparison to NL-means
Application: Point Cloud Compression


Self-similarity for compression

[Hubo et al. 2008]
- Cluster surface patches by similarity
- Replace each patch by a word of the codebook

Compression for rendering and not precision!

Patch-based self-similarity
Local patches capture local variations, comparing them underlines the self-similarity
Two samplings of the same shape
Pipeline

Original Seeds and patches Parameterization

Patch descriptions Coefficients Dictionary

1 2 n-1 n

Beyond classification: Dictionary Learning
Working assumptions

- **Topological condition**: Surface covered by a set of topological disks centered around seeds.

- **Sampling condition**: $R$-neighborhood of a seed containing enough points.

- **Noise level**: Noise magnitude strictly below radius $R$.

- **Seeds selection**: anchors to define local patches
Self-similarity compression

- **Seeds** selection
- Local patches represented in a comparable way
- Patches decomposed upon a dictionary found by the K-SVD algorithm
- Final data: a set of seeds with local frames, a small dictionary and the (sparse) coefficients for each patch.
Further compression

- **Seeds**: kd-tree compression [Gandoin and Devillers, 2002].
- **Local parameterization** (3 Euler angles): predictive coding
- **Dictionary**: lossless compression.
- **Coefficients**: scalar quantization (increases sparsity) followed by entropy coding.
Controlling the error

Two types of errors are introduced:

- **Resampling error**
  ⇒ *Increasing the accuracy of the resampling pattern*

- **Compression error**
  ⇒ *Increasing the number of atoms in the dictionary*
Decompression

1. Decompress
   - seed positions
   - euler angles
   - dictionary $D$
   - coefficients $A$

2. Reconstruct the patches:
   $$P_{rec} = D \times A$$

3. Consolidate the reconstructed point cloud in overlapping areas.
Results

Anubis (9, 9M pts) compressed to 0.96bpp; error = 0.01mm (0.003%)
St Matthew (93,5M pts) compressed to 0.83bpp; error = 0.05cm (0.002%)
Mire (16,1M pts) compressed to 0.73bpp; error = 0.03mm (0.011%). Screened Poisson Reconstruction [Kazdhan, 2013]
Comparison with kd-tree coding. 4.83bpp against 0.6bpp in our method.
Lovers (15, 8M pts) compressed to 0.59 bpp; error = 0.01mm (0.006%)
Breaking the working assumptions

Church (69, 9M pts) compressed to 0.76bpp; error = 1.48cm (0.005%)
rate/distortion performance compared to previous works
Outline

1 Introduction
2 General Formulation
3 Support Vector Machine
4 On Trees and Forests
5 Boosting
6 Beyond classification: Dictionary Learning
7 Neural nets
8 Generative problems
9 Generative Adversarial Networks (GAN)
Neural Nets

- Net of *Neurons* organized into *layers*
- When a net has many layers it is called *deep*.
- Meta-parameters (number of layers...) and learnable parameters
- Disclaimer: we are not going to address the link between neural nets and actual human brain neurons

A useful tool

Neural Net playground https://playground.tensorflow.org/
Neuron model

Neuron

A *linear operation* of the inputs followed by an *activation function*
A linear operation of the inputs followed by an activation function.

- Linear operation outputs: $o_1 = x_1 \times w_1 + x_2 \times w_2 + b$
- Activation function. Output: $o_2 = f(o_1)$
Activation function

$f$ can be a non linear function. Typical activation functions:

- **Sigmoid function** $f(x) = \frac{1}{1+e^{-x}}$
- **Hyperbolic Tangent** $f(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}}$
- **Rectified Linear Unit (ReLU)** $f(x) = \max(0, x)$
Forward pass

- Given values $x$ compute the output value
- $w_1 = 1$, $w_2 = 2$, $b = 1$, $f$ ReLU. For $x_1 = 0.5$, $x_2 = -0.5$
  
  $\text{output} = \max(0, 1 \times 0.5 + 2 \times (-0.5) + 1) = 0.5$

- $w_1 = 1$, $w_2 = 2$, $b = 1$, $f$ ReLU. For $x_1 = 0.5$, $x_2 = -2$
  
  $\text{output} = \max(0, 1 \times 0.5 + 2 \times (-2) + 1) = 0$
Net model

- Input layer - 1 hidden layer - output layer
- Fully connected net: each neuron from a layer is connected to all neurons of the next layer.
- Forward pass $y = \text{some}_\text{function}(x_1, x_2)$
Training a neural net

- **Loss function** for the training dataset we know what values we want for $y$. Measure the difference:

$$MSE(y, y_{true}) = (y - y_{pred})^2$$

- Can be seen as a function:

$$L(x_1, x_2, w_{11}, w_{12}, b_{11}, b_{12}, w_{21}, w_{22}, b_{21}, y_{true})$$

- Parameters computation by Gradient Descent

$$w_{11} = w_{11} - \nu \frac{\partial^2 L}{\partial w_{11}^2}$$
Backpropagation

Backprop

Compute the derivative of $L$ wrt $w_{11}$.

- Key ingredient: Chain rule!
Backpropagation

**Backprop**

Compute the derivative of $L$ wrt $w_{11}$.

- Key ingredient: Chain rule!
- Activation functions can be nondifferentiable (e.g. ReLU).
Generative problems
Generative Problems

**Goal**

Given a set of samples $x_1, x_2, \ldots, x_n$ (images, signals, animations...) learn a model $p_\theta(x)$ of the true underlying distribution $p(x)$. 
Generative Problems

Goal

Given a set of samples $x_1, x_2, \ldots, x_n$ (images, signals, animations...) learn a model $p_\theta(x)$ of the true underlying distribution $p(x)$.

- In practice, we use some prior knowledge of the problem to model $p_\theta$. 
Generative Problems

Goal

Given a set of samples $x_1, x_2, \ldots, x_n$ (images, signals, animations...) learn a model $p_\theta(x)$ of the true underlying distribution $p(x)$.

- In practice, we use some prior knowledge of the problem to model $p_\theta$.
- Optimize $\theta$, to minimize the difference between $p$ and $p_\theta$. 
A completely different approach

Idea

Completely different approach: Use a pretrained CNN (ImageNet) and make the features resemble those of the target image.
A completely different approach

**Idea**

Completely different approach: Use a pretrained CNN (ImageNet) and make the features resemble those of the target image.

Applications to:
- Texture synthesis [Gatys et al. 2015]
A completely different approach

Idea

Completely different approach: Use a pretrained CNN (ImageNet) and make the features resemble those of the target image.

Applications to:

- Texture synthesis [Gatys et al. 2015]
- Style transfer [Gatys et al. 2016].
Style Transfer [Gatys et al. 2016]

A

B

C

D

E

F
Autoregressive maximum likelihood methods (PixelRNN, PixelCNN) [Van der Oord et al. 2016]

Idea
Find the model with the highest likelihood to have generated the data.
Autoregressive maximum likelihood methods (PixelRNN, PixelCNN) [Van der Oord et al. 2016]

Idea

Find the model with the highest likelihood to have generated the data.

- Pixels $x_1, x_2, \ldots, x_n$
Autoregressive maximum likelihood methods (PixelRNN, PixelCNN) [Van der Oord et al. 2016]

Idea

Find the model with the highest likelihood to have generated the data.

- Pixels $x_1, x_2, \cdots, x_n$
- $p_\theta(x_i) = \prod_i p(x_i|x_{i-1}, \cdots, x_n)$
Autoregressive maximum likelihood methods (PixelRNN, PixelCNN) [Van der Oord et al. 2016]

**Idea**
Find the model with the highest likelihood to have generated the data.

- Pixels $x_1, x_2, \ldots, x_n$
- $p_\theta(x_i) = \prod_i p(x_i|x_{i-1}, \ldots, x_n)$

**Process**
Generate pixels sequentially starting from a corner. Dependency on the previous pixels modeled by a Recurrent Neural Network (PixelRNN) or a Convolutional Neural Network (PixelCNN).
PixelRNN

Samples trained on ImageNet, 64x64 images, Image from [Van der Oord et al. 2016].

Pros and Cons

Pros: explicit model of $p_\theta$, Good evaluation metric
Cons: slow because of sequential generation
A reminder on auto-encoders

Goal

Given input data $x$ produce $z$ smaller than $x$ that sums up $x$. 

Training done by encoding $x$ into $z$, decoding $z$ into $\hat{x}$ and minimizing $\parallel x - \hat{x} \parallel^2$. 

Latent space capture data variations Generate new data from a samples in the latent space! 

After training throw away the encoder.
A reminder on auto-encoders

Goal
Given input data $x$ produce $z$ smaller than $x$ that sums up $x$.

- Training done by encoding $x$ into $z$, decoding $z$ into $\hat{x}$ and minimizing $\|x - \hat{x}\|^2$. 

Image copyright Arden Dertat.
A reminder on auto-encoders

Goal

Given input data $x$ produce $z$ smaller than $x$ that sums up $x$.

- Training done by encoding $x$ into $z$, decoding $z$ into $\hat{x}$ and minimizing $\|x - \hat{x}\|^2$.
- Latent space capture data variations Generate new data from a samples in the latent space!
A reminder on auto-encoders

Goal

Given input data $x$ produce $z$ smaller than $x$ that sums up $x$.

- **Training** done by encoding $x$ into $z$, decoding $z$ into $\hat{x}$ and minimizing $\|x - \hat{x}\|^2$.
- **Latent space capture data variations** Generate new data from a samples in the latent space!
A reminder on auto-encoders

Goal
Given input data $x$ produce $z$ smaller than $x$ that sums up $x$.

- Training done by encoding $x$ into $z$, decoding $z$ into $\hat{x}$ and minimizing $\|x - \hat{x}\|^2$.
- Latent space capture data variations Generate new data from a samples in the latent space!
- After training throw away the encoder.
Latent space

Problem

The latent space is not necessarily continuous.
Variational Auto-encoder [Kingma and Welling 2016]

Idea

Ensure that the latent space is continuous.
Variational Auto-encoder [Kingma and Welling 2016]

**Idea**

Ensure that the latent space is continuous. Encoder outputs a standard deviation and a mean instead of a vector.
Variational Auto-Encoder (VAE)

- Add an additional encoder $q_\phi(z|x)$ approximating $p_\theta(z|x)$
Variational Auto-Encoder (VAE)

- Add an additional encoder $q_\phi(z|x)$ approximating $p_\theta(z|x)$
- Encoder Network: $q_\phi(z|x)$, gaussian model: $\mu_{z|x}, \Sigma_{z|x}$, can sample $z|x$ (probabilistic encoder).
Variational Auto-Encoder (VAE)

- Add an additional encoder $q_\phi(z|x)$ approximating $p_\theta(z|x)$
- Encoder Network: $q_\phi(z|x)$, gaussian model: $\mu_{z|x}, \Sigma_{z|x}$, can sample $z|x$ (probabilistic encoder).
- Decoder Network: $p_\theta(x|z)$, gaussian model: $\mu_{x|z}, \Sigma_{x|z}$, can sample $x|z$ (probabilistic decoder).
Variational Auto-Encoder (VAE)

- Add an additional encoder $q_\phi(z|x)$ approximating $p_\theta(z|x)$
- Encoder Network: $q_\phi(z|x)$, gaussian model: $\mu_{z|x}, \Sigma_{z|x}$, can sample $z|x$ (probabilistic encoder).
- Decoder Network: $p_\theta(x|z)$, gaussian model: $\mu_{x|z}, \Sigma_{x|z}$, can sample $x|z$ (probabilistic decoder).
Objective function in a VAE

Minimization

Computing parameters $\theta$, $\phi$ maximizing:

$$L(x_i, \theta, \phi) = \log p_\theta(x_i) \geq E_{z \sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] - D_{KL}(q_\phi(z|x_i)\|p_\theta(z))$$
Objective function in a VAE

Minimization

Computing parameters $\theta$, $\phi$ maximizing:
$$\mathcal{L}(x_i, \theta, \phi) = \log p_\theta(x_i) \geq E_{z \sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] - D_{KL}(q_\phi(z|x_i) \parallel p_\theta(z))$$

- $\mathcal{L}(x_i, \theta, \phi)$ is a lower bound of $p_\theta(x_i)$
Image generation using VAE

- Sample $z$ from gaussian prior (diagonal covariance).

Remark: Diagonal covariance for $z$ yields independent latent variables corresponding to interpretable factors of variation.

Pros & Cons

Pros: Interpolation possible in latent space. Latent variables can be interpretable.

Cons: Maximizes a lower bound of the likelihood, blurry results.
Image generation using VAE

- Sample $z$ from gaussian prior (diagonal covariance).
- Run $z$ through the decoder, yielding $\mu_x|z$ and $\Sigma_x|z$.

Remark: Diagonal covariance for $z$ yields independent latent variables corresponding to interpretable factors of variation.

Pros & Cons

Pros: Interpolation possible in latent space. Latent variables can be interpretable.

Cons: Maximizes a lower bound of the likelihood, blurry results.
Image generation using VAE

- Sample $z$ from gaussian prior (diagonal covariance).
- Run $z$ through the decoder, yielding $\mu_x|z$ and $\Sigma_x|z$.
- Sample $\hat{x}$ from $\mathcal{N}(\mu_x|z, \Sigma_x|z)$. 

Remark: Diagonal covariance for $z$ yields independent latent variables corresponding to interpretable factors of variation.

Pros & Cons
- Pros: Interpolation possible in latent space. Latent variables can be interpretable.
- Cons: Maximizes a lower bound of the likelihood, blurry results.
Image generation using VAE

- Sample $z$ from gaussian prior (diagonal covariance).
- Run $z$ through the decoder, yielding $\mu_x | z$ and $\Sigma_x | z$.
- Sample $\hat{x}$ from $\mathcal{N}(\mu_x | z, \Sigma_x | z)$.

**Remark**

Diagonal covariance for $z$ yields independent latent variables corresponding to interpretable factors of variation.
Image generation using VAE

- Sample $z$ from gaussian prior (diagonal covariance).
- Run $z$ through the decoder, yielding $\mu_x|z$ and $\Sigma_x|z$.
- Sample $\hat{x}$ from $\mathcal{N}(\mu_x|z, \Sigma_x|z)$.

**Remark**

Diagonal covariance for $z$ yields independent latent variables corresponding to interpretable factors of variation.

**Pros & Cons**

**Pros:** Interpolation possible in latent space. Latent variables can be interpretable.

**Cons:** Maximizes a lower bound of the likelihood, blurry results.
VAE Applications

\[ z \sim \mathcal{N}(0, 1) \]

Hou et al. 2016

Generative problems
Outline

1 Introduction
2 General Formulation
3 Support Vector Machine
4 On Trees and Forests
5 Boosting
6 Beyond classification: Dictionary Learning
7 Neural nets
8 Generative problems
9 Generative Adversarial Networks (GAN)
We are not going to model explicitly the density $p_{\theta}(x)$

But we will be able to sample from it!

Sample from a simple distribution and learn the transform to the training distribution.

GAN Principle
We are not going to model explicitly the density $p_\theta(x)$

But we will be able to sample from it!

Sample from a simple distribution and learn the transform to the training distribution.

---

**Generative Adversarial training**

Admit you have an oracle $D$ that rates if an image $I$ looks *real* ($D(I) = 1$) or *unreal* ($D(I) = 0$). If you want to synthesize an image, you want this oracle to judge the synthesized image as *real*.
GAN Principle

- We are not going to model explicitly the density $p_\theta(x)$
- But we will be able to sample from it!
- Sample from a simple distribution and learn the transform to the training distribution.

Generative Adversarial training

Admit you have an oracle $D$ that rates if an image $I$ looks real ($D(I) = 1$) or unreal ($D(I) = 0$). If you want to synthesize an image, you want this oracle to judge the synthesized image as real.

- Sadly, we have no oracle $D$ available.
GAN

2 players Game

G tries to synthesize images that will fool D and D tries to distinguish between real images and fake images synthesized by G.
2 players Game

$G$ tries to synthesize images that will fool $D$ and $D$ tries to distinguish between real images and fake images synthesized by $G$.

Objective Function

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x \sim p_{data}(x)} [\log D_{\theta_D}(x)] + \mathbb{E}_{z \sim p_{prior}(z)} [\log (1 - D_{\theta_D}(G_{\theta_G}(z)))]$$

Where $\theta_D$ (resp. $\theta_G$) are the parameters of the discriminator (resp. generator).
GAN training

Alternate optimization

Alternate between

1. Optimize parameters $\theta_D$ by gradient ascent ($\theta_G$ fixed).
2. Optimize parameters $\theta_G$ by gradient descent ($\theta_D$ fixed).
GAN training

Alternate optimization

Alternate between

1. Optimize parameters $\theta_D$ by gradient ascent ($\theta_G$ fixed).
2. Optimize parameters $\theta_G$ by gradient descent ($\theta_D$ fixed).

Do we need all the terms of the objective functions for the two steps?
GAN training

Alternate optimization

Alternate between
1. Optimize parameters $\theta_D$ by gradient ascent ($\theta_G$ fixed).
2. Optimize parameters $\theta_G$ by gradient descent ($\theta_D$ fixed).

Do we need all the terms of the objective functions for the two steps?

Problem

In practice hard to optimize! Alternative:
1. Optimize parameters $\theta_D$ by gradient ascent ($\theta_G$ fixed).
2. Optimize parameters $\theta_G$ by gradient ascent ($\theta_D$ fixed) with objective:

$$\max_{\theta_G} \mathbb{E}_{z \sim p_{\text{prior}}(z)} \log D_{\theta_D}(G_{\theta_G}(z))$$
**Training Algorithm**

**Algorithm 1: Training**

1. **for** $j = 1 \cdots N$ **do**
   2. **for** $k = 1 \cdots K$ **do**
      3. Sample a minibatch of $m$ samples $z_i$;
      4. Sample a minibatch of $m$ real samples $x_i$;
      5. Update $\theta_D$:
         \[
         \theta_D = \theta_D + \nu \nabla_{\theta_D} \left( \sum_{i=1}^{m} \log D_{\theta_D}(x_i) + \log(1 - D_{\theta_D}(G_{\theta_G}(z_i))) \right)
         \]
      6. Sample a minibatch of $m$ samples $z_i$;
      7. Update $\theta_G$:
         \[
         \theta_G = \theta_G + \nu \nabla_{\theta_G} \left( \sum_{i=1}^{m} \log(D_{\theta_D}(G_{\theta_G}(z_i))) \right)
         \]
Training Algorithm

Algorithm 2: Training

for $j = 1 \cdots N$ do
	for $k = 1 \cdots K$ do
		Sample a minibatch of $m$ samples $z_i$;
		Sample a minibatch of $m$ real samples $x_i$;
		Update $\theta_D$:
		$$\theta_D = \theta_D + \nu \nabla_{\theta_D} \left( \sum_{i=1}^{m} \log D_{\theta_D}(x_i) + \log(1 - D_{\theta_D}(G_{\theta_G}(z_i))) \right)$$
	Sample a minibatch of $m$ samples $z_i$;
	Update $\theta_G$:
	$$\theta_G = \theta_G + \nu \nabla_{\theta_G} \left( \sum_{i=1}^{m} \log(D_{\theta_D}(G_{\theta_G}(z_i))) \right)$$

Generation

Sample $z$ and generate $\hat{x} = G(z)$. 

Generative Adversarial Networks (GAN)
Training Algorithm

Algorithm 3: Training

for $j = 1 \cdots N$ do
  for $k = 1 \cdots K$ do
    Sample a minibatch of $m$ samples $z_i$;
    Sample a minibatch of $m$ real samples $x_i$;
    Update $\theta_D$:
    $$\theta_D = \theta_D + \nu \nabla_{\theta_D} \left( \sum_{i=1}^{m} \log D_{\theta_D}(x_i) + \log(1 - D_{\theta_D}(G_{\theta_G}(z_i))) \right)$$
  
  Sample a minibatch of $m$ samples $z_i$;
  Update $\theta_G$:
  $$\theta_G = \theta_G + \nu \nabla_{\theta_G} \left( \sum_{i=1}^{m} \log(D_{\theta_D}(G_{\theta_G}(z_i))) \right)$$
end for

Generation

Sample $z$ and generate $\hat{x} = G(z)$. $D$ is not needed.
Results
What are $D$ and $G$?

Deep convolutional GANs

[Radford et al. 2016]
GAN analysis

Pros and Cons

Pros: State-of-the-art results, difficult to quantify the quality of the results.
Cons: Difficult to train, cannot produce the explicit density.

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

[Radford et al. 2016]
Latent space arithmetic

[Radford et al. 2016]
Latent space arithmetic

[Radford et al. 2016]
Comparison: pixel space arithmetic

[Radford et al. 2016]
Conditional GANs

cGAN idea

Condition $G$ and $D$ on some additional variable $y$. Feed $y$ to both $G$ and $D$.

Mirza et al. 2014
Conditional GANs

cGAN idea

Condition $G$ and $D$ on some additional variable $y$. Feed $y$ to both $G$ and $D$.

Objective Function

$$\min_G \max_D E_{x \sim p_{data}}[\log D(x|y)] + E_{z \sim p_{prior}}[(1 - \log D(G(z)|y))]$$
### Results of conditional GAN

<table>
<thead>
<tr>
<th>User tags + annotations</th>
<th>Generated tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car</td>
<td>taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails</td>
</tr>
<tr>
<td>food, raspberry, delicious, homemade</td>
<td>chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes</td>
</tr>
<tr>
<td>water, river</td>
<td>creek, lake, along, near, river, rocky, treeline, valley, woods, waters</td>
</tr>
<tr>
<td>people, portrait, female, baby, indoor</td>
<td>love, people, posing, girl, young, strangers, pretty, women, happy, life</td>
</tr>
</tbody>
</table>

[Mirza et al. 2014]
Conditioning on images

**Image-to-Image Translation [Isola et al. 2017]**

\[ y \text{ is now an image we want to transform (sketch to object, day to night, B/W to color...). Other formulation:} \]

\[
\min_G \max_D \mathbb{E}_{(x,y) \sim p_{\text{data}}} \left[ \log D(x, y) \right] + \mathbb{E}_{y \sim p_{\text{data}}, z \sim p_{\text{prior}}} \left[ (1 - \log D(G(z, y) | y)) \right] \\
+ \lambda \mathbb{E}_{x, y, z} \left[ \| x - G(z, y) \|_1 \right]
\]

- Additional term favors resemblance to true result and produces better results [Pathak et al. 2014]
Conditioning on images

[Isola et al. 2017]
Conclusion

- Many more applications of Machine Learning to investigate
- Machine Learning:
  - Irregularly sampled data? Geometry?
- Several open questions on Deep Learning
  - Deep Learning: Why does it work?
  - Given a problem, find a rule to design the network?
Some reading