Adaptivity and Self-Supervision for Mobile Robots

Raia Hadsell

Carnegie Mellon: Martial Hebert, Drew Bagnell, Daniel Huber

New York University: Yann LeCun, Pierre Sermanet, Ayse Erkan

Sarnoff Corporation: Teddy Kumar, Harpreet Sawney
Adaptivity

- Flexibility and adaptability – indicator of intelligence
- Capabilities are developed in response to new experiences and new environments
  - **Navigating** through an unknown world
  - **Communicating** with aliens
  - **Moving** or manipulating in novel ways
- Adaptability implies learning, but learning
  - Within a *temporal framework*
  - Through **interaction and validation** rather than supervision
Adaptivity

How should adaptivity be studied?
Adaptivity

- How should adaptivity be studied?
Adaptivity

How should adaptivity be studied?
Adaptivity

- Robots are the best subjects
  - We want them to perform complex, higher-order tasks
    - Navigation
    - Conversation
    - walking/running/soccer
  - They come with multiple sensors
    - Cameras
    - LIDAR
    - proprioceptive
  - They interact with the real world
Outline

- 3 projects
- 3 architectures for adaptivity
- **No** conclusions
Task: 3D reconstruction of rough terrain
Proposed Solution

1. Learn a surface function using kernel regression
   - **Continuous**, not discrete

2. Use Visibility information
   - “Space-Carving” - visibility of points constrains surface
     - **Points** must lie on surface (positive information)
     - **Rays** connecting sensor and points must lie above surface
Proposed Solution

1. Learn a surface function using kernel regression
   - **Continuous**, not discrete

2. Use Visibility information
   - “Space-Carving” - visibility of points constrains surface
     - Points must lie on surface (positive information)
     - Rays connecting sensor and points must lie above surface

3. Estimate **upper and lower bounds** simultaneously

4. Optimize learning with stochastic gradient-based algorithm
   - New data can be added **online** and surface will adapt
   - Allows **anytime processing**: surface estimate and bounds always available
Data – single linear constraint

$z = f(x)$
Data – single linear constraint

\[ z = f(x) \]
Visibility – infinite linear constraints

\[ z = f(x) \]
Visibility – infinite linear constraints

\[ z = f(x) \]
Visibility – infinite linear constraints

\[ z = f(x) \]

Given \[ S = \{ (x_1, z_1, ray_1), (x_2, z_2, ray_2), \ldots, (x_n, z_n, ray_n) \} \]

find \[ f : \mathbb{R}^2 \rightarrow \mathbb{R} \]

s.t. \[ f(x_i) = z_i \quad \forall x_i, \]

\[ f(x) \leq ray_i(x) \quad \forall s_i, x \]
Rough Terrain Evaluation

- Dataset collected with Velodyne and cameras on Boss
- Uneven distribution of points
- 100,000 training set
- 40x40 meter region

1K pts/.5m²
Rough Terrain Evaluation - Surface
Rough Terrain Evaluation – Upper bound
Rough Terrain Evaluation – Lower bound
Rough Terrain – Cross Section 2

Cross Section of Terrain with Uncertainty Bounds (Boss-rough5, x=-14.0)
The Lengthscale dilemma
The Lengthscale dilemma

The **lengthscale** of the kernel basis functions is critical:

- **Small lengthscale**: higher precision, danger of overfitting
- **Large lengthscale**: lower precision, avoids fitting to noise
Adaptive solution: Learn lengthscale slowly

Adapt lengthscale based on visual confirmation of smoothness of terrain, plus proprioceptive feedback from vehicle and local data density and variance

Observations:
- Image features (gradient, texture)
- Vehicle response (odometry, wheel encod)
- Local data density and variance

\[
L_{vis} = f_{vis}(I(x, y))
\]
\[
L_{odo} = f_{odo}(V_{t-k}, \ldots, V_t)
\]
\[
L_D = f_D(D_N(x, y, z))
\]
\[
L = f_W(L_{vis}, L_{odo}, L_D)
\]
Long Range Vision on a LAGR Robot
LAGR (Learning Applied to Ground Robots)

- Long-range vision and mapping implemented on LAGR platform: LAGR (Learning Applied to Ground Robots): DARPA 2005-08
- 10 research labs develop learning and vision algorithms

- NREC designed hardware and baseline software:
  - 2 stereo color camera pairs
  - GPS receiver for global navigation
  - 2 front bumper switches
  - Onboard IMU (inertial measurement unit)
  - 4 onboard Linux computers

- **Goal:** Navigation in unknown terrain using long-range vision
The Problem

- Long-range perception is difficult, even for humans
  - lighting, low resolutions, ambiguous objects, occlusion, etc.
- But learning from multiple sensory inputs improves vision:
  - close-range sensors (reliable) correct long-range sensors
The Problem

- Long-range perception is difficult, even for humans
  - lighting, low resolutions, ambiguous objects, occlusion, etc.
- But learning from multiple sensory inputs improves vision:
  - close-range sensors (reliable) correct long-range sensors
The Problem

- Long-range perception is difficult, even for humans
  - lighting, low resolutions, ambiguous objects, occlusion, etc.
- But learning from multiple sensory inputs improves vision:
  - close-range sensors (reliable) correct long-range sensors
The Problem

- Long-range perception is difficult, even for humans
  - lighting, low resolutions, ambiguous objects, occlusion, etc.
- But learning from multiple sensory inputs improves vision:
  - close-range sensors (reliable) correct long-range sensors

A mobile robot equipped with short and long-range sensors could learn in a similar way.
A Long Range Learning Strategy

Our approach: Near-to-Far, Self-supervised online learning

(A) Labels from a short-range sensor (stereo 3d)
(B) Train a classifier using labels + short-range image data
(C) Classify entire image – short-range and long-range
A Long Range Learning Strategy

Our approach: Near-to-Far, Self-supervised online learning

(A) Labels from a short-range sensor (stereo 3d)
(B) Train a classifier using labels + short-range image data
(C) Classify entire image – short-range and long-range

Input images (stereo 512x384)
A Long Range Learning Strategy

- Our approach: Near-to-Far, Self-supervised online learning
  - (A) Labels from a short-range sensor (stereo 3d)
  - (B) Train a classifier using labels + short-range image data
  - (C) Classify entire image – short-range and long-range
A Long Range Learning Strategy

Our approach: Near-to-Far, Self-supervised online learning

(A) Labels from a short-range sensor (stereo 3d)
(B) Train a classifier using labels + short-range image data
(C) Classify entire image – short-range and long-range
A Long Range Learning Strategy

Our approach: Near-to-Far, Self-supervised online learning

- (A) Labels from a short-range sensor (stereo 3d)
- (B) Train a classifier using labels + short-range image data
- (C) Classify entire image – short-range and long-range
A Long Range Learning Strategy

Our approach: Near-to-Far, Self-supervised online learning

- (A) Labels from a short-range sensor (stereo 3d)
- (B) Train a classifier using labels + short-range image data
- (C) Classify entire image – short-range and long-range
Unsupervised Deep Feature Learning: Convolutional Autoencoder

$Z_2$ (features/code)

Train second layer using reconstruction criteria

$Z_1$ (features/code)

Train first layer using reconstruction criteria

Input window
Self-supervision from stereo pointcloud

Label windows using stereo information

- super-ground
- ground
- footline
- obstacle
- super-obstacle
Online Training and Classification

**Loss:**

\[
L_{nll} = \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i W_i' D_i}\right)
\]

**Gradient:**

\[
\frac{\partial L}{\partial W} = -\left(\frac{y + 1}{2} - g_w(X)\right)X
\]

- 750 samples of each class are kept in a ring buffer: short term memory.
- Learning “snaps” to new environment in about 10 frames
- Weights are trained with stochastic gradient descent
- Regularization by decay to default weights

\[
g_w(X) = \frac{1}{1 + \exp(W_i' X)}
\]

**Feature Extractor**

**Logistic Regression**

\[
X: 100 \times 1
\]

\[
\text{Label from stereo}
\]
The full architecture

Input window

Information rich; large and contextual
The full architecture

(Unsupervised) Convolutional neural network

Trained with unsupervised learning; adapts very slowly (or not at all) to fundamental changes in environment

Input window
The full architecture

(Supervised) Multi-class logistic regression

(Unsupervised) Convolutional neural network

Input window

$\mathbf{f}_W(\mathbf{x})$

Self-supervised classifier adapts quickly and forgets quickly
The full architecture

Likelihood vector

(Supervised) Multi-class logistic regression

(Unsupervised) Convolutional neural network

Input window

In theory the output should also be rich and contextual
The full architecture

- Input window
- (Unsupervised) Convolutional neural network
- (Supervised) Multi-class logistic regression
- Likelihood vector
- $f_w(x)$

Self-supervised fast learning

Unsupervised slow learning
Long Range Vision Results

Input image  | Stereo Labels  | Classifier Output
---|---|---

Input image  | Stereo Labels  | Classifier Output

Input image  | Stereo Labels  | Classifier Output

Input image  | Stereo Labels  | Classifier Output
Field Tests

left camera

right camera

Cost Map (FastOD & FarOD)

stereo only

Cost Map (FastOD & FarOD)

long range
Field Tests

left camera

right camera

stereo only

long range
Field Tests

left camera

right camera

stereo only

long range
start

with learning

no learning

intervention

goal
(Sarnoff) Audio-visual sensor fusion on mobile robots
Acoustic or contact microphone gives local audio signal as the vehicle's wheels roll over the ground surface.
(Sarnoff) Audio-visual sensor fusion on mobile robots

- Acoustic or contact microphone gives local audio signal as the vehicle's wheels roll over the ground surface
- Cameras give mid and long range visual information
Audio-visual sensor fusion on mobile robots

- Acoustic or contact microphone gives local audio signal as the vehicle's wheels roll over the ground surface.
- Cameras give mid and long range visual information.
- Sensor fusion transfers knowledge and reliability between the two sensors through **transfer learning** and **cross validation**.
  - Sensor range is increased.
  - Reliability is increased.
  - Accurate recognition is increased.
(Sarnoff) Audio-visual sensor fusion on mobile robots

- **Time** $t$: Vehicle records visual input for location $(x,y)$
- Accurate localization is required
(Sarnoff) Audio-visual sensor fusion on mobile robots

- **Time** $t$: Vehicle records visual input for location $(x,y)$
  - Accurate localization is required
- **Time** $t+n$: Vehicle records audio signal for location $(x,y)$
(Sarnoff) Audio-visual sensor fusion on mobile robots

- **Time** $t$: Vehicle records visual input for location $(x,y)$
- **Time** $t+n$: Vehicle records audio signal for location $(x,y)$

Visual + audio signals are fused with a classifier to a common representation.
A classifier or regression can fuse the audio and visual sensors.

EM-type algorithms are well-suited for sensor fusion:
- First the audio processing function is fixed and the parameterized visual function $g(x)$ is optimized,
- Then the visual processing function is fixed and the parameterized audio function $f(x)$ is optimized.
A classifier or regression can fuse the audio and visual sensors. EM-type algorithms are well-suited for sensor fusion:

- First the audio processing function is fixed and the parameterized visual function $g(x)$ is optimized,
- Then the visual processing function is fixed and the parameterized audio function $f(x)$ is optimized.

$$y = h(x_a, x_v)$$
A classifier or regression can fuse the audio and visual sensors.

- **EM-type algorithms** are well-suited for sensor fusion:
  - First the audio processing function is fixed and the parameterized visual function $g(x)$ is optimized,
  - Then the visual processing function is fixed and the parameterized audio function $f(x)$ is optimized.

Sensor fusion and prediction:

$$y = h(x_a, x_v)$$

Emphasis in this architecture is on cross-validation and learning through knowledge transfer and verification rather than layered fast/slow learning.
Conclusions

Robust adaptation depends on multiple kinds of learning co-existing in complex architectures:

- Supervised
- Self-supervised
- Unsupervised
- Fast/slow
- Shallow/deep

Challenges:

- Optimization (non-convex, non-linear architectures)
- Memory (what to remember? what to forget?)
Thanks

Martial Hebert
Drew Bagnell
Daniel Huber

Yann LeCun
Ayse Erkan
Pierre Sermanet
Matt Grimes
Marc'Aurelio Ranzato
Koray Kavukcuoglu
Sumit Chopra

Urs Muller
Chris Crudele
Marco Scoffier

Tyler Neylon
Ryan McDonald
Casey Whitelaw

Jan Ben

Larry Jackel
Eric Krotkov
Shallow vs. Deep Learning

- **Shallow/Local architectures (e.g., kernel methods)**
  - Fixed layer of kernel functions match the input to templates extracted from the training data; output is a linear combination of the matching score.
  - Convex for some loss functions, domain-specific pre-processing, local, poor extrapolation beyond the training set.
Shallow vs. Deep Learning

- Deep architectures (e.g., multi-layer RBMs, Convolutional nets)
- Cascade of non-linear, trainable, parameterized modules
- Depth allows representation of complex functions in a more compact form (depth-breadth trade-off)
- Optimization is never convex
- Each module adds increasing abstraction to pattern recognition
Deep architectures (e.g., multi-layer RBMs, Convolutional nets)
Cascade of non-linear, trainable, parameterized modules
Depth allows representation of complex functions in a more compact form (depth-breadth trade-off)
Optimization is never convex
Each module adds increasing abstraction to pattern recognition
Shallow vs. Deep Learning

- Deep architectures (e.g., multi-layer RBMs, Convolutional nets)
- Cascade of non-linear, trainable, parameterized modules
- Depth allows representation of complex functions in a more compact form (depth-breadth trade-off)
- Optimization is never convex
- Each module adds increasing abstraction to pattern recognition
Shallow vs. Deep Learning

- Deep architectures (e.g., multi-layer RBMs, Convolutional nets)
- Cascade of non-linear, trainable, parameterized modules
- Depth allows representation of complex functions in a more compact form (depth-breadth trade-off)
- Optimization is never convex
- Each module adds increasing abstraction to pattern recognition
Shallow vs. Deep Learning

- Deep architectures (e.g., multi-layer RBMs, Convolutional nets)
- Cascade of non-linear, trainable, parameterized modules
- Depth allows representation of complex functions in a more compact form (depth-breadth trade-off)
- Optimization is never convex
- Each module adds increasing abstraction to pattern recognition
Shallow vs. Deep Learning

- Deep architectures (e.g., multi-layer RBMs, Convolutional nets)
- Cascade of non-linear, trainable, parameterized modules
- Depth allows representation of complex functions in a more compact form (depth-breadth trade-off)
- Optimization is never convex
- Each module adds increasing abstraction to pattern recognition
Deep architectures (e.g., multi-layer RBMs, Convolutional nets)

Cascade of non-linear, trainable, parameterized modules

Depth allows representation of complex functions in a more compact form (depth-breadth trade-off)

Optimization is never convex

Each module adds increasing abstraction to pattern recognition