Artificial Intelligence
V08: Learning Agents

Introduction to supervised machine learning
Decision trees
Doing machine learning

Based on material by
• Stuart Russell, UC Berkeley
• Andreas Krause, ETH Zurich
Educational objectives

• **Remember** the basic **decision tree training algorithm**
• **Explain** machine learning using the correct **technical terms**
• **Defend** your **own view on** the existence of good **general learners**
• **Build** decision tree-based **models** for labeled data sets using the **ML development process**

*In which we describe agents that can improve their behaviour through diligent study of their own experiences.*

➔ Reading: AIMA, ch. 18-18.6
1. INTRODUCTION TO SUPERVISED MACHINE LEARNING

- **Training phase**
  - Performance standard
    - Critic
      - feedback
        - learning goals
          - Problem generator
            - often replaced by fixed training set
- **Labels**
  - Sensors
    - changes
      - knowledge
        - Actuators
          - testing / application phase
  - Environment

1.1. Training
1.2. Test
The discipline of machine learning – mapped
«…gives computers the ability to learn without being explicitly programmed.»
A. Samuel, 1959

Famous: used in most human learning, definition of scientific method

Types of:
- inductive (example-based)
- transductive (example → example)
- deductive (logic-based)

Subtypes of:
- supervised (learn concepts / predict values)
- reinforcement (learn to act)
- unsupervised (find structure)

Models:
- linear / non-linear
- parametric / non-parametric
- discriminative / generative
- predictive / inferential
- deep / shallow

Search through hypothesis space, driven by theory and empiricism
Supervised machine learning in a nutshell

Training data points, represented by some feature vector $x$
Supervised machine learning in a nutshell

Training data points, represented by some feature vector $x$
Supervised machine learning in a nutshell

Training data points, represented by some feature vector $\mathbf{x}$

We hope (and design) for good generalization to unseen test data
Supervised machine learning in a nutshell

**Training data** points, represented by some **feature vector** \( \mathbf{x} \)

*This model is probably overfitting the training data*

*We hope (and design) for good generalization to unseen test data*
Supervised machine learning in a nutshell

We search for models (functions) in a hypothesis space $\mathcal{H}$ by minimizing loss $\ell$ between label $y$ and result $h(x)$.

Training data points, represented by some feature vector $x$.

This model seems neither to overfit nor underfit.

We hope (and design) for good generalization to unseen test data.
Learning as search through $\mathcal{H}$

\[
\mathcal{H} = \{ \}
\]
Learning as search through $\mathcal{H}$

$\mathcal{H} = \{ / \mid \ldots \}$
Learning as search through $\mathcal{H}$

$\mathcal{H} = \{ (\text{...}) \}$

Success is largely determined by **choosing the correct hypothesis space** for the problem:
- Linear? Polynomial?
- Deep neural network? CNN?
- Ensemble of decision trees? …

$$h(x) = h(x, w)$$

Learning then means finding good **parameters** (sometimes called $\theta$)
Learning as search through $H$

Success is largely determined by choosing the correct hypothesis space for the problem:
- Linear? Polynomial?
- Deep neural network? CNN?
- Ensemble of decision trees? …

$h(x) = h(x, w)$

Learning then means finding good parameters (sometimes called $\theta$)
Learning as search through $\mathcal{H}$

$\mathcal{H} = \{ h(x) = h(x, w) \}$

Success is largely determined by **choosing the correct hypothesis space** for the problem:
- Linear? Polynomial?
- Deep neural network? CNN?
- Ensemble of decision trees? …

Learning then means finding good **parameters** (sometimes called $\theta$)
Learning as search through $\mathcal{H}$

$$\mathcal{H} = \{ \}$$

Success is largely determined by choosing the correct hypothesis space for the problem:
- Linear? Polynomial?
- Deep neural network? CNN?
- Ensemble of decision trees? …

$$h(x) = h(x, w)$$

A good model complies with Ockham's razor: Maximize a combination of consistency and simplicity

Learning then means finding good parameters (sometimes called $\theta$)
What is this current hype about deep learning?
Add depth (layers $\Rightarrow$ capability) to learn features automatically

Classic computer vision

Feature extraction
(SIFT, SURF, LBP, HOG, etc.)

Classification
(SVM, neuronal net, etc.)

Takes raw pixels as input, learns good features automatically!

Convolutional neural networks (CNNs)
Why study machine learning in general?

«A learner that makes *no a priori assumptions* regarding the identity of the target concept has *no rational basis for classifying* any unseen instances»

[Mitchell, 1997, ch. 2.7.3]

There’s no single best algorithm

• **No free lunch theorem** (NFL) regarding the general equivalence of learners [Wolpert, 1996]:
  When all hypotheses $h$ are equally likely, the probability of observing an arbitrary sequence of cost values during training does not depend upon the learning algorithm $L$  
  ➔ **there’s no universally best learner** (across problems)

• **Empirical study** [Caruana et al., 2006]:
  «Even the best models sometimes perform poorly, and models with poor average performance occasionally perform exceptionally well»
  ➔ **All learning algorithms have advantages & disadvantages, depending on the current data**

Ascertainment from kaggle.com

• **Tabular** data: do *handcrafted* feature engineering, followed by an *ensemble of decision trees*

• **Sensor** data (images, speech, …): use a *suitable deep neural network*
Why is there no universally best learner? 
Even if not, can there be a good general learner?

ML research unanimously states that “there is no universally best learner”. But a general learner doesn’t need to work for all possible kinds of data – it may suffice that it works well on all data relevant to human problem solving.

- [Optional] Conduct a quick search: What does the NFL theorem really claim (and what not)?
- Conduct a quick search on the concept of the “inductive bias” of a learning algorithm as its brought-in prior knowledge (e.g. Tom Mitchell’s work)
- Discuss: Are there more general forms of prior knowledge that universally guide learning?
2. DECISION TREES
Attribute-based representations of data
Valid for all kinds of data ( , )

Examples described by features
• Possible attribute values: Boolean, discrete, continuous, etc.
• Example: “Situations where I will/won't **wait for a table**”

- Alternative nearby?
- Has a bar to wait in?
- Is it Friday?
- Really hungry?
- How crowded already?
- Raining outside?
- Did make reservation?
- Minutes to wait

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>F</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₂</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
</tr>
<tr>
<td>X₃</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₄</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
</tr>
<tr>
<td>X₅</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X₆</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₇</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X₈</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₉</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X₁₀</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
</tr>
<tr>
<td>X₁₁</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X₁₂</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
</tr>
</tbody>
</table>

• Goal: **classification** of examples into positive (T) or negative (F) class
Attribute-based representations of data
Valid for all kinds of data (⋯⋯)

Examples described by features
• Possible attribute values: Boolean, discrete, continuous, etc.
• Example: “Situations where I will/won't wait for a table”

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hub</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₂</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
</tr>
<tr>
<td>X₃</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₄</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
</tr>
<tr>
<td>X₅</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X₆</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₇</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X₈</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₉</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X₁₀</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
</tr>
<tr>
<td>X₁₁</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X₁₂</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
</tr>
</tbody>
</table>

• Goal: classification of examples into positive (T) or negative (F) class
Decision tree representation of hypotheses

Example: Stuart Russell’s “true” tree to decide whether to wait in a restaurant

Expressiveness

- **Decision trees** can **express any function** of the input attributes
  E.g. for Boolean functions: truth table row → path to leaf
- **Trivial** tree ∀ training sets: **one path** to leaf for each example
  But probably won’t generalize to new examples
  ➔ Prefer to find more **compact** decision trees
Hypothesis spaces

Even a constrained hypothesis space is large
- **How many distinct** decision trees with \( n \) Boolean attributes?
  - \( = \) number of Boolean functions
  - \( = \) number of distinct truth tables with \( 2^n \) rows = \( 2^{2n} \)
  - Example: 6 Boolean attributes \( \Rightarrow 18'446'744'073'709'551'616 \) possible trees

- How many purely conjunctive hypotheses (e.g., \( Hungry \land \neg Rain \))
  - Each attribute can be either positive, negative, or out of the hypothesis
  \( \Rightarrow 3^n \)

More expressive hypothesis spaces
- ...increase chance that target function can be **expressed 😊**
- ...increases **number** of hypotheses consistent w/ training set
  \( \Rightarrow \text{may get worse} \) predictions 😞

Due to overfitting we have seen earlier
Decision tree learning

Goal: find a small tree consistent with the training examples
Idea: (recursively) choose “most significant” attribute as root of (sub)tree

Algorithm

- function LearnDecisionTree(examples, attributes) returns a tree
  return DecisionTreeLearning(examples, attributes, [])

function DecisionTreeLearning(examples, attributes, parent_examples) returns a tree
  if examples is empty then return PluralityValue(parent_examples)
  else if all examples have the same classification then return the classification
  else if attributes is empty then return PluralityValue(examples)
  else
    A ← argmax_{a ∈ attributes} Importance(a, examples)
    tree ← a new decision tree with root test A
    for each value v_k of A do #for categorical features
      exs ← {e: e ∈ examples and e.A = v_k}
      subtree ← DecisionTreeLearning(exs, attributes - A, examples)
      add a branch to tree with label (A = v_k) and subtree subtree
    return tree

- PluralityValue(examples) selects the most common output among examples
- Importance(attribute, examples) selects the most important attribute
- On ties, both functions choose randomly
Choosing an attribute
How to implement Importance(attribute, examples)

Idea: A **good attribute splits** examples into subsets that are (ideally) “all pos” or “all neg”

Example

- **Patrons** is better choice: gives information about the classification

Recap: Information theory

- **Information answers questions**: The more cluelessness an observation removes, the more information it contains
- Inversely proportional to entropy (uncertainty of a random variable)
  - A Boolean answer with prior $< 0.5$, $0.5 >$ has entropy $= 1$ bit (if we remove this uncertainty, we gain 1 bit of info.)
  - A coin giving heads 99$\%$ of the time has entropy close to 0 ($\approx 0.08$ bits $\rightarrow$ almost no info.-gain when observed)
  - Entropy in an observation (having prior $< P_1, ..., P_n >$): $H(P_1, ..., P_n) = -\sum_{i=1}^{n} P_i \log_2 P_i$
Information gain as splitting criterion

Suppose we have \( p \) positive and \( n \) negative examples at the root

- \( H\left( \left( \frac{p}{p+n}, \frac{n}{p+n} \right) \right) \) bits needed to classify a new example
- E.g., for the 12 restaurant examples, \( p = n = 6 \), so we need overall 1 bit

An attribute \( A \) splits the examples \( E \) into subsets \( E_i \) (one per possible value)

- Each of which (we hope) needs less information to complete the classification
- Let \( E_i \) have \( p_i \) positive and \( n_i \) negative examples
  \[ H\left( \left( \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i} \right) \right) \] bits needed to classify a new example
- Expected number of necessary bits per example over all branches \( i \) stemming from \( A \) is

\[
\text{Remainder}(A) = \sum_i \frac{p_i + n_i}{p + n} H\left( \left( \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i} \right) \right) \]

- For Patrons this is 0.459 bits, for Type this is (still) 1 bit
  \[ \Rightarrow \text{Choose} \text{ the attribute that minimizes the remaining information needed, …} \]
  \[ \Rightarrow \text{i.e., maximizes information gain:} \ G(A) = H\left( \left( \frac{p}{p+n}, \frac{n}{p+n} \right) \right) - \text{Remainder}(A) \]
The learned decision tree
Based on our 12 examples

- Substantially simpler than "true" tree
  → E.g., Reservation and Raining are not needed (perfect classification possible without)
- A more complex hypothesis isn't justified by the small amount of data
  → But what makes one tree better than another?
3. DOING MACHINE LEARNING

[Diagram showing the process of machine learning with stages such as preprocessing, feature extraction, modeling, and recognition.]
Performance measurement
The ML development process being an empirical science

Hume’s “Problem of Induction” (1740): when is generalization admissible?

How do we know that \( h \approx f \) (the true function)?
1. Use theorems of computational/statistical learning theory
2. Try \( h \) on a new test set of examples
   - Prerequisite for inductive learning: generalizes (only) to same distribution as seen in training set!
   - Best practice: use cross-validation to train & validate on different sets before final test
   - Test set (ca. 20%)

3. Report performance using recognized figures of merit
   - E.g. accuracy (or test set error) if all errors are equally costly: \[ \text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \]
   - E.g. recall/precision if false alarms and misses differ in cost: \[ \text{recall} = \frac{TP}{TP + FN}, \quad \text{precision} = \frac{TP}{TP + FP} \]
   - Conduct repeatable experiments (i.e., fully scriptable, full documentation of inputs and results)

| classification → |
|↓ label          |
| 1               |
| 0               |
| 1               |
| true positive (TP, “hit”) | false negative (FN, “miss”) |
| 0               |
| false positive (FP, “false alarm”) | true negative (TN) |
Debugging machine learning models

Learning curve: % correct on train & test set as a function of training set size
• Diagnostic: reveals over- and underfitting as well as realizability (→ see appendix)

What to try next when a given model generalizes poorly?
• Get more training examples → fixes overfitting
• Try smaller sets of features → fixes overfitting
• Try getting additional features → fixes underfitting
• Try adding polynomial features \(x_1, x_2, x_1^2, x_2^2, \ldots\) → fixes underfitting
• Try less regularization → fixes underfitting
• Try more regularization → fixes overfitting
• Build ensembles → fixes overfitting, uses limited data best (→ see V09)

Regularization: Any method that limits the expressiveness of the hypothesis space by adding constraints to learning; e.g., pruning decision trees.
Where’s the intelligence?  
Man vs. machine

- Machine learning offers **general function approximations purely learned** from examples
- But: **Success depends on** a good fit of the algorithm’s inductive bias to problem at hand  
  → i.e., **clever algorithm choice** based on experience

- Learning is a **powerful principle of self-optimization, applicable to all** components of previously seen agent designs
- But: **General** (domain crossing, knowledge-linking) **learning must be** based on way better inclusion of **unsupervised** learning principles (besides general inductive biases)  
  → current avant-garde deep learning research explores this route (→ see e.g. GANs in V11)

- **Decision trees** in principle are simple models (appreciated for their simplicity in formalism and interpretation), suitable only for Excel-like data
- But: **Combining multiple trees** (called an “ensemble”) makes them **extremely powerful** for all but **pattern recognition** (i.e., sensor data-based) problems  
  (and sometimes even there → see V09)
Review

- Learning needed for unknown environments, “lazy designers”
- Learning agent = performance element (testing / application phase) + learning element (training phase)
- **Learning method** (algorithm) depends on…
  - type of performance element (classify? regress? control?),
  - available feedback (labels),
  - type of component to be improved (representation? utility function? action?),
  - and data representation (numerical or categorical data, logical clauses, raw pixels, …)
- For supervised learning, the aim is to find a simple hypothesis that is approximately consistent with training examples and generalizes well
- **Decision tree** learning uses information gain
  - Popular models because of easy interpretability
  - Many famous implementations (e.g. CART, C4.5®)
  - As ensembles: **very good general-purpose out-of-the-box models** (e.g. Random Forest®, XGBoost → see V09)
- Learning performance = prediction accuracy measured on separate test set
  - Development using 5-fold cross validation (without ever looking at test set!)
  - Systematic and repeatable experiments are paramount (e.g. using UNIX-style scripts)
APPENDIX
Learning curves
Diagnosing learning problems

Learning curve, simplified: %correct on test set only as a function of training set size

Accuracy shown in learning curve depends on
- **Realizability** (target function expressible in chosen hypothesis space?)
  - Non-realizability can be due to **missing attributes**
  - or **restricted hypothesis class** (e.g., a thresholded linear function might be overly simplistic)
- **Redundant** features
  (e.g., loads of irrelevant attributes make learning difficult)
Why is this current hype about deep learning?
The ImageNet Competition (more on deep learning → see appendix)

A. Krizhevsky uses a «Deep Convolutional Neural Network» (CNN) for the first time
Why is this current hype about deep learning?
The ImageNet Competition (more on deep learning → see appendix)

1000 categories
1 mio. training examples

A. Krizhevsky uses a «Deep Convolutional Neural Network» (CNN) for the first time
Why is this current hype about deep learning?
The ImageNet Competition (more on deep learning → see appendix)

1000 categories
1 mio. training examples
...

2015: Computers learned to «see»
4.95% Microsoft (Feb 06)
→ super-human performance (human: 5.10%)
4.80% Google (Feb 11)
4.58% Baidu (May 11)
3.57% Microsoft (Dec 10)

A. Krizhevsky uses a «Deep Convolutional Neural Network» (CNN) for the first time
2016: A summer of breakthroughs in ML
...enabled by deep learning

Impressive novelties within a summer’s timespan
• Game playing: beating the human Go world champion
• Audio synthesis: Synthesizing speech & music sample by sample
• Art style transfer: Redraw the content of a picture in the style of any painting
• Image synthesis: Completion of missing parts in pictures
• Text synthesis: Generation of text in specific styles (e.g., Shakespeare, $L^A T_E X$, …)
• Word vectors: Arithmetic with semantic meaning of text and images

→ See next slides
Google will buy London-based artificial intelligence company DeepMind. The Information reports that the acquisition price was more than $500 million, and that Facebook was also in talks to buy the startup late last year. DeepMind confirmed the acquisition to us, but couldn't disclose deal terms.

The acquisition was originally confirmed by Google to Re/code.
Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

Posted Jan 26, 2014 by Catherine Shu (@catherineshu)

Google will buy DeepMind, an artificial intelligence startup, reports that the company is in talks to buy the company, which couldn't disclose deal terms.

The acquisition was originally confirmed by Google to Re/code.
Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

Posted Jan 26, 2014 by Catherine Shu (@catherineshu)

Google will buy DeepMind for a reported $500 million, reports that the company is in talks to buy AlphaGo, which it couldn't disclose deal terms.

The acquisition was originally confirmed by Google to Re/code.
Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

Posted Jan 26, 2014 by Catherine Shu (@catherineshu)

Google will buy DeepMind, reports that the company is in talks to buy Absinthe, and that the deal couldn’t disclose deal terms.

The acquisition was originally confirmed by Google to Re/code.

40 days

AlphaGo Zero surpasses all other versions of AlphaGo and, arguably, becomes the best Go player in the world. It does this entirely from self-play, with no human intervention and using no historical data.
Google’s WaveNet uses neural nets to generate eerily convincing speech and music

Posted Sep 9, 2016 by Devin Coldewey

Generating speech from a piece of text is a common and important task undertaken by computers, but it’s pretty rare that the result could be mistaken for ordinary speech. A new technique from researchers at Alphabet’s DeepMind takes a completely different approach, producing speech and even music that sounds eerily like the real thing.

Early systems used a large library of the parts of speech (phonemes and morphemes) and a large ruleset that described all the ways letters combined to produce those sounds. The pieces were joined, or concatenated, creating functional speech synthesis that can handle most words, albeit with unconvincing cadence and tone. Later systems parameterized the generation of sound, making a library of speech fragments unnecessary. More compact — but often less effective.

**WaveNet, as the system is called,** takes things deeper. It simulates the sound of speech at as low a level as possible: one sample at a time. That means building the waveform from scratch — 16,000 samples per second.
Google’s WaveNet uses neural nets to generate eerily convincing speech and music

Intro

What if you could imitate a famous celebrity’s voice or sing like a famous singer? This project started with a goal to convert someone’s voice to a specific target voice. So called, it’s voice style transfer. We worked on this project that aims to convert someone’s voice to a famous English actress Kate Winslet’s voice. We implemented a deep neural networks to achieve that and more than 2 hours of audio book sentences read by Kate Winslet are used as a dataset.

Model Architecture

This is a many-to-one voice conversion system. The main significance of this work is that we could generate a target speaker’s utterances without parallel data like <source’s way, target’s way>, <way, text> or <way, phone>, but only waveforms of the target speaker. (To make these parallel datasets needs a lot of effort.) All we need in this project is a number of waveforms of the target speaker’s utterances and only a small set of <way, phone> pairs from a number of anonymous speakers.

WaveNet, as the system is called, takes as input a wav file and produces wav output. The underlying assumptions are that sound is a waveform, allowing a single sample to come from a wide range of sources. By starting with an input wav file and generating a corresponding output wav file, we are essentially creating a prototype for a new medium—speech synthesis for the Web.

Generating speech from a piece of text requires understanding of the language being spoken, which comes from a large ruleset that described all the ways in which one word can be classified. As individual pieces were joined, or concatenated, creating new words, the new words were almost always meaningless, but often less effective.

With WaveNet, we are able to take a wav file and produce a corresponding wav file, which can then be used as input to a speech recognition system. This allows us to use the system in a wide range of applications, from voice-activated assistants to text-to-speech systems.
Computing

Algorithm Clones Van Gogh’s Artistic Style and Pastes It onto Other Images, Movies

A deep neural network has learned to transfer artistic styles to other images.

by Emerging Technology from the arXiv May 10, 2016

The nature of artistic style is something of a mystery to most people. Think of Vincent Van Gogh’s Starry Night, Picasso’s work on cubism, or Edvard Munch’s The Scream. All have a powerful, unique style that humans recognize easily.
Computing

Algorithm Clones Van Gogh’s Artistic Style and Pastes It onto Other Images, Movies

A deep neural network has learned to transfer artistic styles to other images.

by Emerging Technology from the arXiv  May 10, 2016

The nature of artistic style is something of a mystery to most people. Think of Vincent Van Gogh’s Starry Night, or Edvard Munch’s The Scream. Humans can recognize these images, but no one knows exactly what makes them special.
Computing

Algorithm Clones Van Gogh’s Artistic Style and Pastes It onto Other Images, Movies

A deep neural network has learned to transfer artistic styles to other images.

by Emerging Technology from the arXiv May 10, 2016

The nature of artistic style is something of a mystery to most people. Think of Vincent Van Gogh’s Starry Night or Edvard Munch’s The Scream, or Claude Monet’s Water Lilies. Do these paintings reflect the artist’s individual vision, or do humans recognize easily.
Computing

Algorithm Clones Van Gogh’s Artistic Style and Pastes It onto Other Images, Movies

A deep neural network has learned to transfer artistic styles to other images.

by Emerging Technology from the arXiv May 10, 2016

The nature of artistic style is something of a mystery to most people. Think of Vincent Van Gogh’s Starry Night, or Edvard Munch’s The Scream. These images are iconic, and humans recognize easily.
Deep neural networks can now transfer the style of one photo onto another

And the results are impressive

by James Vincent | @jvincent | Mar 30, 2017, 1:53pm EDT

A deep neural network can copy the style of another image.

by Emerging Tech

The nature of art
of Vincent Van Gogh, Edvard Munch’s son, humans recognize

Original photo | Reference photo | Result

You’ve probably heard of an AI technique known as “style transfer” — or, if you haven’t heard of it, you’ve seen it. The process uses neural networks to apply the look and feel of one image to another, and appears in apps like Prisma and Facebook. These style transfers, however, are stylistic, not photorealistic. They look good because they look like they’ve been painted. Now a group of researchers from Cornell University and Adobe have augmented
...and the list could be continued
...and the list could be continued
...and the list could be continued
...and the list could be continued

Image Completion with Deep Learning in TensorFlow

- Introduction
  - Step 1: Interpreting images as samples from a probability distribution
    - How would you fill in the missing information?
    - But where do statistics fit in? These are images.
    - So how can we complete images?
  - Step 2: Qualifying generating fake images
    - Learning to generate new samples from an unknown probability distribution
    - (DL-Herz) Generate adversarial 2048 (GAN) building blocks
    - (DL-Herz) To produce fake images
    - [DL-Herz] Training DCGANs
    - Existing GANs
    - (DL-Herz) Running DO
  - Step 3: Finding the
    - Image completion
    - (DL-Herz)
    - (DL-Herz)
    - Completing
    - Conclusion
    - Partial bibliography
    - Bonus: Incompleteness

The amazing power of word vectors

For today's post, I've drawn material not just from one paper, but from five! The subject matter is 'word vector' – the work of Mikolov et al. at Google on efficient vector representations of words (and what you can do with them). The papers are:

- Efficient Estimation of Word Representations in Vector Space – Mikolov et al. 2013
- Distributed Representations of Words and Phrases and their Compositionality – Mikolov et al. 2013
- Linguistic Regularities in Continuous Space Word Representations – Mikolov et al. 2013
- word2vec Parameter Learning Explained – Rong 2014
- word2vec Explained: Deriving Mikolov et al.'s Negative Sampling Word-Embedding Method – Goldberg and Levy 2014

From the first of these papers ('Efficient estimation...') we get a description of the Continuous Bag-of-Words and Continuous Skip-gram models for learning word vectors (we'll talk about what a word vector is in a moment...). From the second paper we get more illustrations of the power of word vectors, some additional information on optimisation for the skip-gram model (hierarchical softmax and negative sampling), and a discussion of the continuous bag-of-words model, which the authors claim is more robust and efficient to train than the skip-gram model.
Inductive supervised learning

Assumption

- A model fit to **enough** training examples...
- ...will **generalize** well to unseen test data
Inductive supervised learning

Assumption
- A model fit to enough training examples…
- …will generalize well to unseen test data

Method
- Search for parameters of a given class of functions…
- …such that every training input (e.g. an image) is mapped to the correct output label (e.g. «car»)

Source: http://lear.inrialpes.fr/job/postdoc-large-scale-classif-11-img/attrs_patchwork.jpg
What is the effect of parameter search?
What is the effect of more capable function classes?

Neuron

Features (e.g. pixels) → Adjustable parameters → Decision (threshold) → Result (e.g. «1» for «car») → Output

\[
\sum \mathbf{w} \mathbf{x} = \mathbf{y}
\]
What is the effect of parameter search?
What is the effect of more capable function classes?

Neuron

Features (e.g. pixels)
Adjustable parameters
Decision (threshold)
Result (e.g. «1» for «car»)
What is the effect of parameter search? What is the effect of more capable function classes?

Neuron

- Inputs
- Features (e.g. pixels)
- Adjustable parameters
- Sum
- Activation function
- Decision (threshold)
- Result (e.g. «1» for «car»)

Neural Network

- Hidden layer (n = 16 neurons)
- Output layer (n = 10 neurons)

Zurich University of Applied Sciences and Arts
InIT Institute of Applied Information Technology (stdm)
How are the parameters found?

- Definition of the neural net: \( f_{\theta}(x) = y \)
  with image \( x \), true result \( y \) and all parameters \( \tilde{\theta} \)
  \( (\tilde{\theta} = \{w_1, w_2\} \text{ chosen randomly at start}) \)

- Error measure: \( J(\tilde{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (f_{\tilde{\theta}}(x_i) - y_i)^2 \)
  Mean squared error
How are the parameters found?

- Definition of the neural net: \( f_{\theta}(x) = y \)
  with image \( x \), true result \( y \) and all parameters \( \theta \)
  \( (\theta = \{w_1, w_2\} \) chosen randomly at start \)

- Error measure: \( J(\theta) = \frac{1}{N} \sum_{i=1}^{N} (f_{\theta}(x_i) - y_i)^2 \)
  Mean squared error

Probability [%] for a specific outcome

\( \phi \)

Error landscape

\( J(\theta_0, \theta_1) \)
How are the parameters found?

- **Definition of the neural net:** \( f_{\hat{\theta}}(x) = y \)
  
  with image \( x \), true result \( y \) and all parameters \( \hat{\theta} \) 
  
  \( \hat{\theta} = \{w_1, w_2\} \) chosen randomly at start

- **Error measure:** 
  
  \[
  J(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (f_{\hat{\theta}}(x_i) - y_i)^2
  \]

  Mean squared error

**Error landscape**

**Method:** Adaptation of weights of \( f \) in the direction of the steepest gradient (descending) of \( J \)
How are the parameters found?

• Definition of the neural net: \( f_{\hat{\theta}}(x) = y \)
  with image \( x \), true result \( y \) and all parameters \( \hat{\theta} \)
  \( (\hat{\theta} = \{w_1, w_2\} \) chosen randomly at start\)

• Error measure: \( J(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (f_{\hat{\theta}}(x_i) - y_i)^2 \)
  Mean squared error

Probability [%] for a specific outcome

Method: Adaptation of weights of \( f \) in the direction of the steepest gradient (descending) of \( J \)
How are the parameters found?

- Definition of the neural net: \( f_{\hat{\theta}}(x) = y \)
  
  with image \( x \), true result \( y \) and all parameters \( \hat{\theta} \)
  
  \( \hat{\theta} = \{w_1, w_2\} \) chosen randomly at start

- Error measure: \( J(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (f_{\hat{\theta}}(x_i) - y_i)^2 \)
  
  Mean squared error

Probability [%] for a specific outcome

\( \theta_0 \) \hspace{1cm} \theta_1 \hspace{1cm} J(\theta_0, \theta_1)

\( \Rightarrow \) Error landscape

Method: Adaptation of weights of \( f \) in the direction of the steepest gradient (descending) of \( J \)
How are the parameters found?

- Definition of the neural net: \( f_{\tilde{\theta}}(x) = y \)
  with image \( x \), true result \( y \) and all parameters \( \tilde{\theta} \)
  \( (\tilde{\theta} = \{w_1, w_2\} \) chosen randomly at start

- Error measure: \( J(\tilde{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (f_{\tilde{\theta}}(x_i) - y_i)^2 \)
  Mean squared error

\[ θ_0 \]  \[ θ_1 \]

\[ J(θ_0, θ_1) \]

\[ J(\theta_0, \theta_1) \]

Method: Adaptation of weights of \( f \) in the direction of the steepest gradient (descending) of \( J \)

Probability [%] for a specific outcome

Error landscape
How are the parameters found?

- Definition of the neural net: \( f_{\hat{\theta}}(x) = y \)
  
  with image \( x \), true result \( y \) and all parameters \( \hat{\theta} \)
  
  \( \hat{\theta} = \{w_1, w_2\} \) chosen randomly at start

- Error measure:
  
  \[
  J(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (f_{\hat{\theta}}(x_i) - y_i)^2
  \]
  
  Mean squared error

**Method:** Adaptation of weights of \( f \) in the direction of the steepest gradient (descending) of \( J \)
How are the parameters found?

- **Definition of the neural net:** \( f_{\bar{\theta}}(x) = y \)
  
  with image \( x \), true result \( y \) and all parameters \( \bar{\theta} \)
  
  \( (\bar{\theta} = \{w_1, w_2\} \text{ chosen randomly at start}) \)

- **Error measure:**
  
  \[
  J(\bar{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (f_{\bar{\theta}}(x_i) - y_i)^2
  \]
  
  Mean squared error

- **Method:** Adaptation of weights of \( f \) in the direction of the steepest gradient (descending) of \( J \)

- **Probability [%] for a specific outcome**
What does a neural network «see»?  
A hierarchy of progressively complex features

Sources: https://www.pinterest.com/explore/artificial-neural-network/  
What does a neural network «see»?
A hierarchy of progressively complex features

Sources: https://www.pinterest.com/explore/artificial-neural-network/
What does a neural network «see»?
A hierarchy of progressively complex features, visualized

Source: http://vision03.csail.mit.edu/cnn_art/data/single_layer.png