Machine Learning for Programming

Peter Norvig

Google

What is Programming?

What is Programming?

- Given a specification of a function *f*

What is Programming?

- Given a specification of a function *f*
- Implement *f* that meets the specification

What is Programming?

- Given a specification of a function *f*
- Implement *f* that meets the specification

Other things too…

What is Machine Learning?

What is Machine Learning?

- Given some example (*x, y*) pairs

What is Machine Learning?

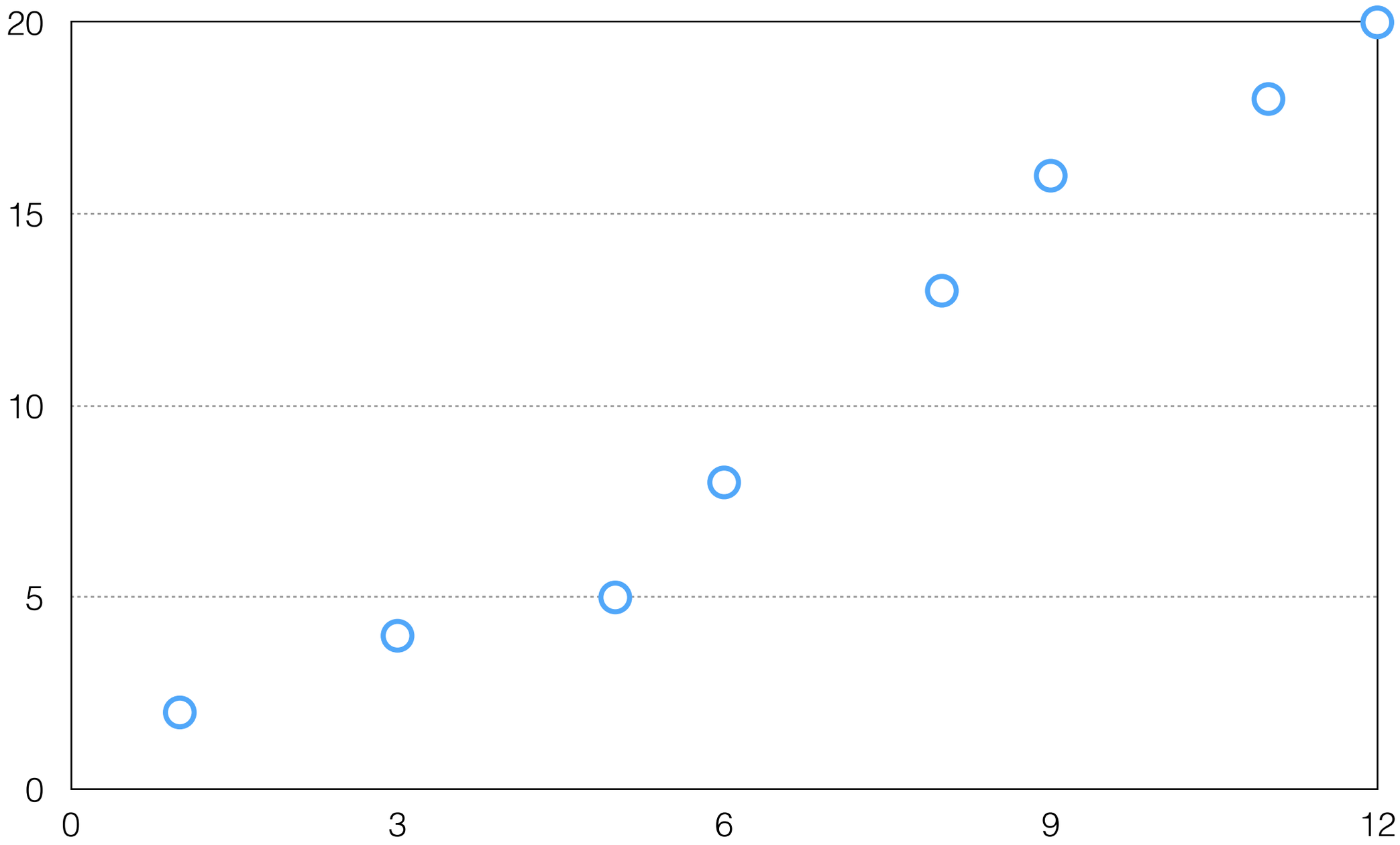
- Given some example (*x*, *y*) pairs
- Induce *f* such that $y \simeq f(x)$,
  for given pairs,
  and generalizes well for unseen *x*

What is Machine Learning?

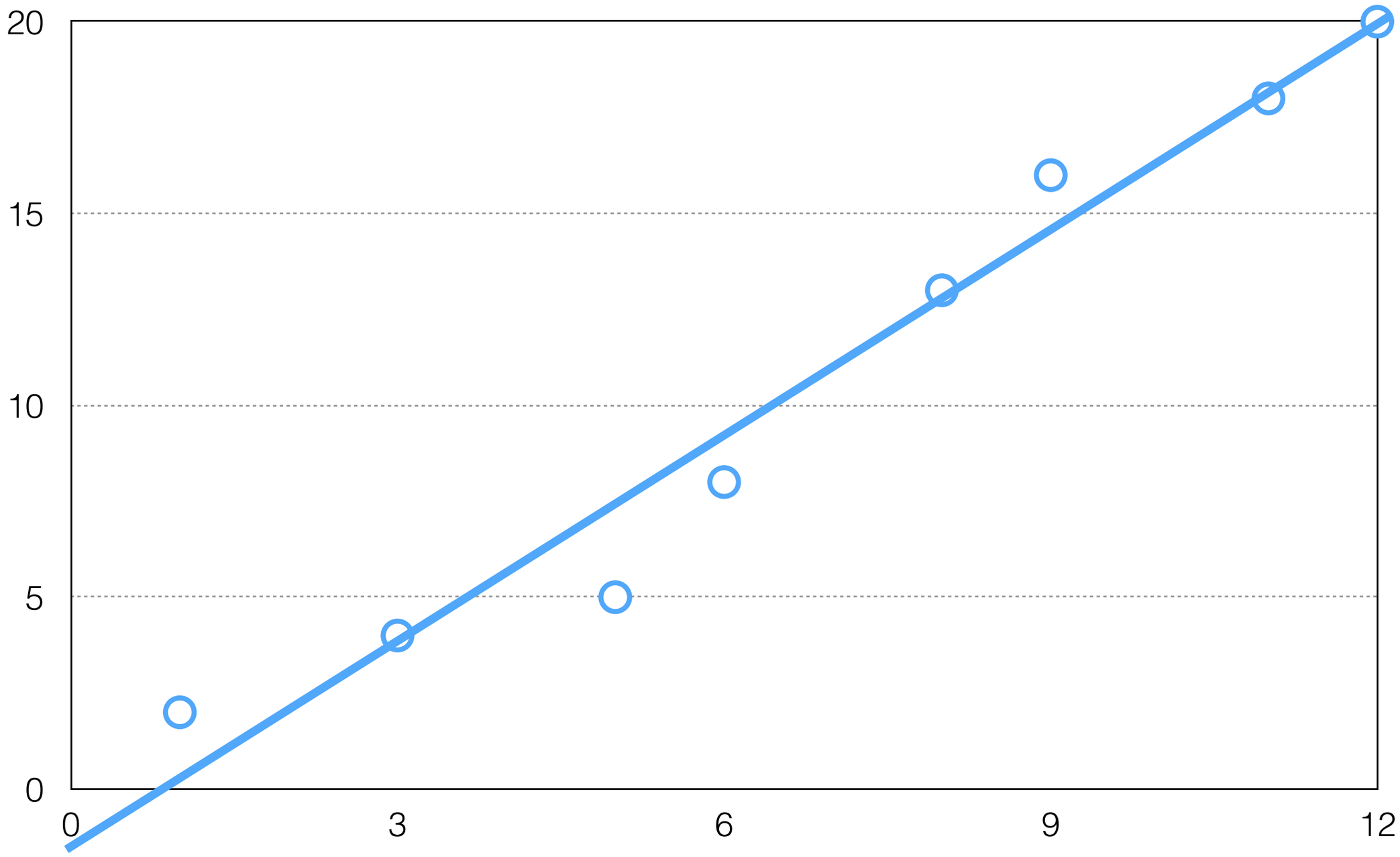
- Given some example (*x*, *y*) pairs
- Induce *f* such that $y \simeq f(x)$,
  for given pairs,
  and generalizes well for unseen *x*

Other kinds of ML too…

Q: Can we learn **functions** from examples?


20
15
10
5
0
0
3
6
9
12

$$y = f(x) = 1.75\,x - 1.2$$



Other (*x*, *y*) pairs

- x = 
  y = “manta ray”

- x = 
  y = “spotted eagle ray”

- x = “This is a short sentence.”
  y = “Esta es una frase corta.”

Q: Can we learn **functions** from examples?

A: **YES**, for many kinds of functions

Q: Can we learn
parts of programs
from examples?

```
import collections, re

def words(text): return re.findall('[a-z]+', text.lower())

WORDS = collections.Counter(words(file('big.txt').read()))
alphabet = 'abcdefghijklmnopqrstuvwxyz'

def edits1(word):
   splits     = [(word[:i], word[i:]) for i in range(len(word) + 1)]
   deletes    = [a + b[1:] for a, b in splits if b]
   transposes = [a + b[1] + b[0] + b[2:] for a, b in splits if len(b)>1]
   replaces   = [a + c + b[1:] for a, b in splits for c in alphabet if b]
   inserts    = [a + c + b     for a, b in splits for c in alphabet]
   return set(deletes + transposes + replaces + inserts)

def known_edits2(word):
   return [e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in WORDS

def known(words): return [w for w in words if w in WORDS]

def correct(word):
   candidates = known([word]) or known(edits1(word)) or known_edits2(word) or [word
   return max(candidates, key=WORDS.get)
```

```
import collections, re

def words(text): return re.findall('[a-z]+', text.lower())

WORDS = collections.Counter(words(file('big.txt').read()))
alphabet = 'abcdefghijklmnopqrstuvwxyz'

def edits1(word):
   splits     = [(word[:i], word[i:]) for i in range(len(word) + 1)]
   deletes    = [a + b[1:] for a, b in splits if b]
   transposes = [a + b[1] + b[0] + b[2:] for a, b in splits if len(b)>1]
   replaces   = [a + c + b[1:] for a, b in splits for c in alphabet if b]
   inserts    = [a + c + b     for a, b in splits for c in alphabet]
   return set(deletes + transposes + replaces + inserts)

def known_edits2(word):
   return [e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in WORDS

def known(words): return [w for w in words if w in WORDS]

def correct(word):
   candidates = known([word]) or known(edits1(word)) or known_edits2(word) or [word
   return max(candidates, key=WORDS.get)
```

17 lines

Metaphone.cc

```
144
145         for (; *n && key.length() < MAXPHONEMELEN; n++)
146         {
147             /* Drop duplicates except for CC */
148             if (*(n - 1) == *n && *n != 'C')
149               continue;
150             /* Check for F J L M N R or first letter vowel */
151             if (same(*n) || *(n - 1) == '\0' && vowel(*n))
152               key << *n;
153             else
154             {
155                 switch (*n)
156                 {
157                 case 'B':
158                     /*
159                      * B unless in -MB
160                      */
161                     if (*(n + 1) || *(n - 1) != 'M')
162                         key << *n;
163                     break;
164                 case 'C':
165                     /*
166                      * X if in -CIA-, -CH- else S if in
167                      * -CI-, -CE-, -CY- else dropped if
168                      * in -SCI-, -SCE-, -SCY- else K
169                      */
170                     if (*(n - 1) != 'S' || !frontv(*(n + 1)))
171                     {
172                         if (*(n + 1) == 'I' && *(n + 2) == 'A')
173                             key << 'X';
174                         else if (frontv(*(n + 1)))
175                             key << 'S';
176                         else if (*(n + 1) == 'H')
177                             key << (((*(n - 1) == '\0' && !vowel(*
178                                       || *(n - 1) == 'S')
179                                     ? 'K' : 'X');
180                         else
181                             key << 'K';
182                     }
```

http://www.htdig.org/ files/ htdig-3.2.0b5.tar.bz2/ htdig-3.2.0b5/ htfuzzy/

Files | Outline[New!]

..
Accents.cc
Accents.h
Endings.cc
Endings.h
EndingsDB.cc
Exact.cc
Exact.h
Fuzzy.cc
Fuzzy.h
Makefile.am
Makefile.in
Makefile.win32
Metaphone.cc
Metaphone.h
Prefix.cc
Prefix.h
Regexp.cc
Regexp.h
Soundex.cc
Soundex.h
Speling.cc
Speling.h
Substring.cc
Substring.h
SuffixEntry.cc
SuffixEntry.h
Synonym.cc
Synonym.h
htfuzzy.cc

Metaphone.cc

```
144
145         for (; *n && key.length() < MAXPHONEMELEN; n++)
146         {
147             /* Drop duplicates except for CC */
148             if (*(n - 1) == *n && *n != 'C')
149               continue;
150             /* Check for F J L M N R or first letter vowel */
151             if (same(*n) || *(n - 1) == '\0' && vowel(*n))
152               key << *n;
153             else
154             {
155                 switch (*n)
156                 {
157                 case 'B':
158                     /*
159                      * B unless in -MB
160                      */
161                     if (*(n + 1) || *(n - 1) != 'M')
162                         key << *n;
163                     break;
164                 case 'C':
165                     /*
166                      * X if in -CIA-, -CH- else S if in
167                      * -CI-, -CE-, -CY- else dropped if
168                      * in -SCI-, -SCE-, -SCY- else K
169                      */
170                     if (*(n - 1) != 'S' || !frontv(*(n + 1)))
171                     {
172                         if (*(n + 1) == 'I' && *(n + 2) == 'A')
173                             key << 'X';
174                         else if (frontv(*(n + 1)))
175                             key << 'S';
176                         else if (*(n + 1) == 'H')
177                             key << (((*(n - 1) == '\0' && !vowel(*
178                                       || *(n - 1) == 'S')
179                                      ? 'K' : 'X');
180                         else
181                             key << 'K';
182                     }
```

2000+ lines

Q: Can we learn
parts of programs
from examples?

A: **Yes,** Machine Learning
is the Ultimate
Agile Programming.

Q: Can we learn
entire programs
from examples?

Program Induction

Program Induction

- $(x, y) = ([3, 2, 1], [1, 2, 3])$
  $(x, y) = ([0], [0])$

Program Induction

- $(x, y) = ([3, 2, 1], [1, 2, 3])$
  $(x, y) = ([0], [0])$

- f_1 = lambda x: f_1(x[1:]) + x[0:1] if x else []
  f_2 = lambda x: [min(x)] + f_2(remove(min(x), x, 1)) if x else []

Program Induction

- $(x, y) = ([3, 2, 1], [1, 2, 3])$
  $(x, y) = ([0], [0])$

- f_1 = lambda x: f_1(x[1:]) + x[0:1] if x else []
  f_2 = lambda x: [min(x)] + f_2(remove(min(x), x, 1)) if x else []

- $(x, y) = ([3, 1, 4, 1, 5, 9], [9, 5, 1, 4, 1, 3])$
  $(x, y) = ([2, 7, 1, 8, 2, 8, 1, 8], [8, 1, 8, 2, 8, 1, 7, 2])$

Program Induction

- (x, y) = ([3, 2, 1], [1, 2, 3])
  (x, y) = ([0], [0])

- f_1 = lambda x: f_1(x[1:]) + x[0:1] if x else []
  f_2 = lambda x: [min(x)] + f_2(remove(min(x), x, 1)) if x else []

- (x, y) = ([3, 1, 4, 1, 5, 9], [9, 5, 1, 4, 1, 3])
  (x, y) = ([2, 7, 1, 8, 2, 8, 1, 8], [8, 1, 8, 2, 8, 1, 7, 2])

- f_1 (*i.e.,* reverse), not f_2 (*i.e.,* sort)

Program Induction

- 1980s: Logical/Functional languages; GP, ILP
  2000s: Probabilistic search rather than logical
  2010s: Deep; LSTM: intermediate representations

- TOOWTDI languages (more or less)
  Only one of “a + 1” or “1 + a” is allowed/pursued

- Stronger type systems

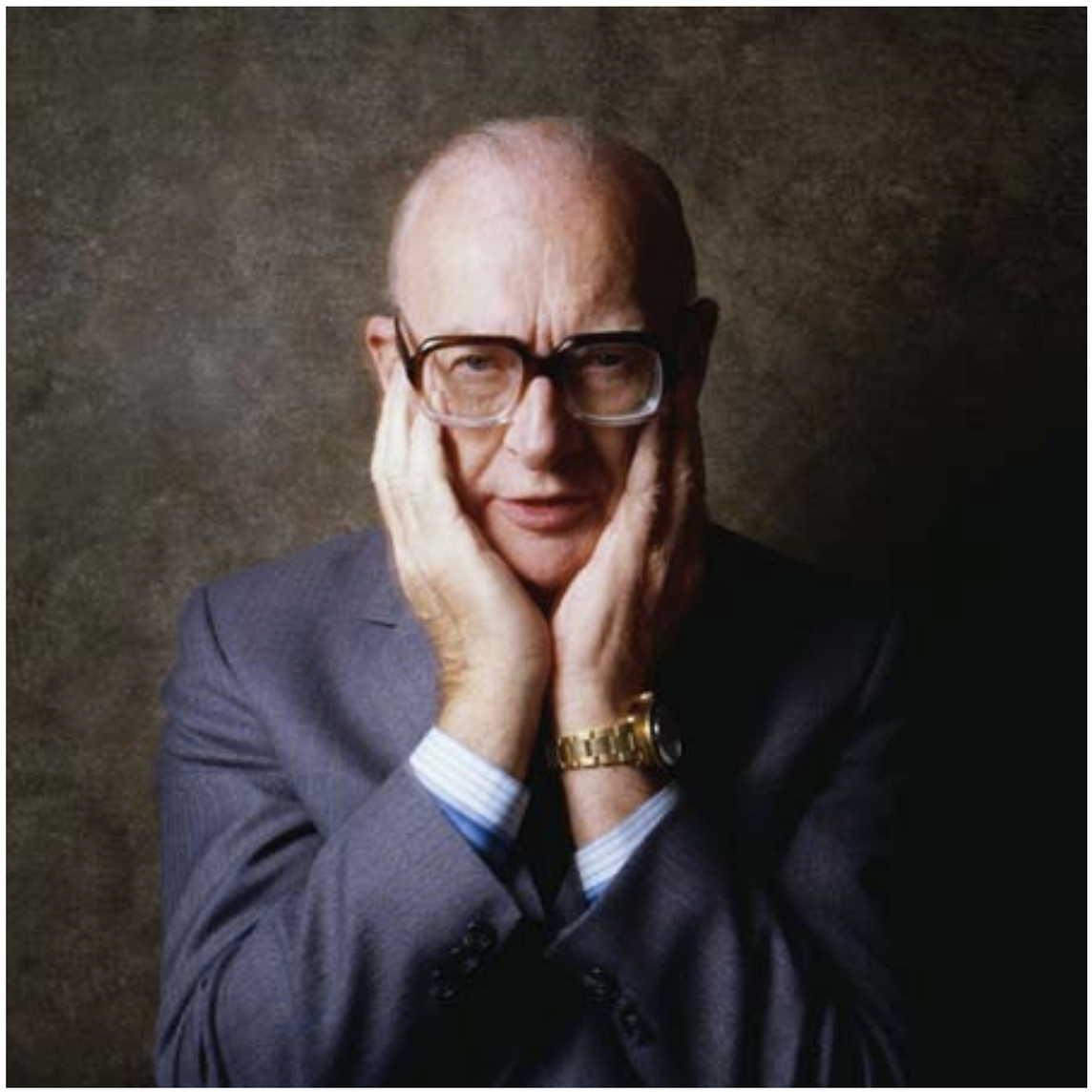
- Total functional programming:
  Only allow recursion that provably terminates

- DSLs, such as regular expressions

Q: Can we learn
entire programs
from examples?

A: **Yes**, for short programs;
No, for complex programs.

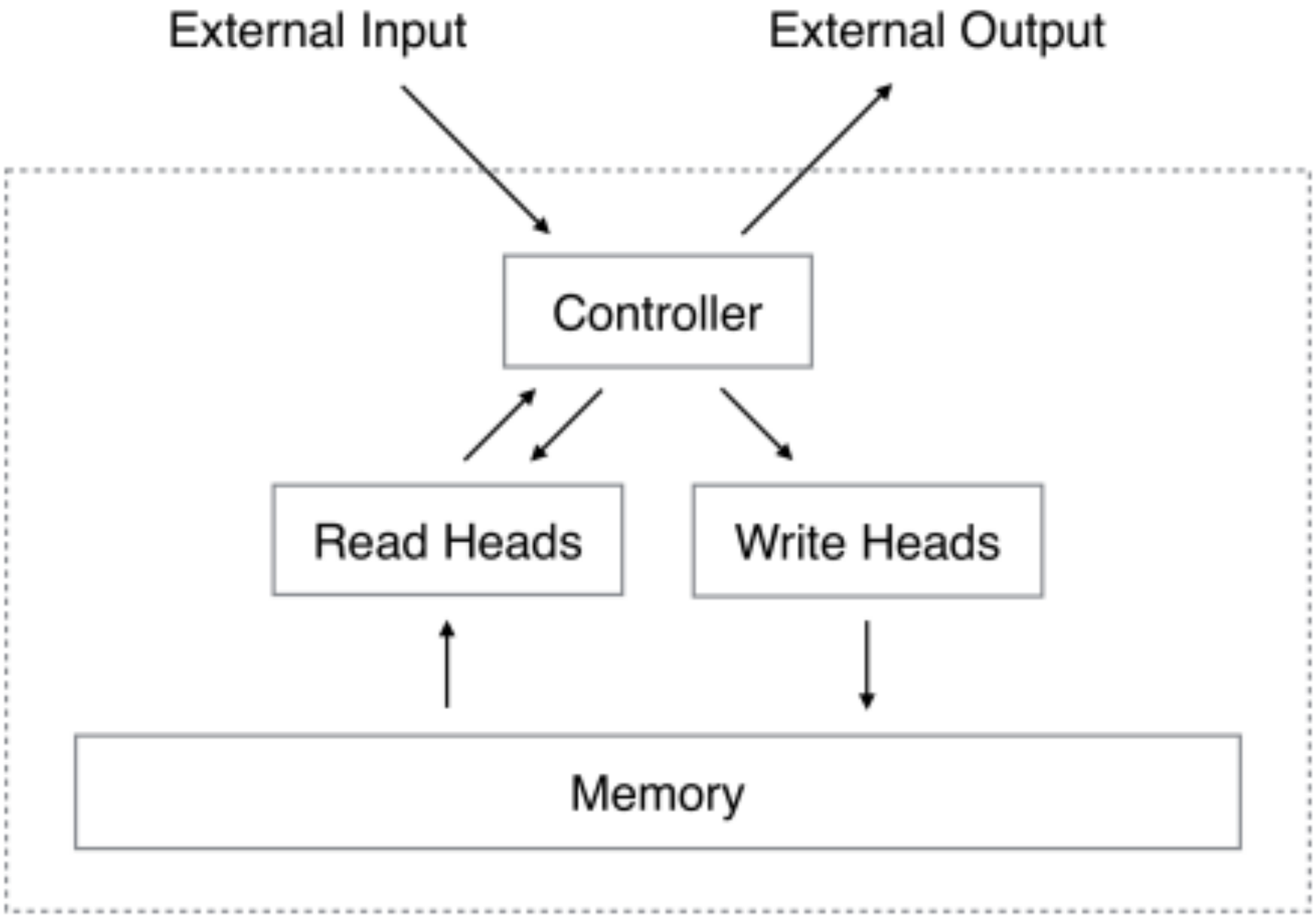
"In the discrete world of computing, there is no meaningful metric in which 'small' changes and 'small' effects go hand in hand, and there never will be."
— Edsger Dijkstra



When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.

— Arthur C. Clarke

Neural Turing Machines



Graves, Wayne, Danihelka

Mini Lesson in ML

- Regression
- Gradient Descent
- Neural Networks

Regression

Regression

- Find a function, f, to minimize
  error on examples,
  and generalization (previously unseen) error

Regression

- Find a function, f, to minimize
  error on examples,
  and generalization (previously unseen) error
- $f = \min_f \text{Loss}(f)$

Regression

- Find a function, f, to minimize
  error on examples,
  and generalization (previously unseen) error

- $f = \min_f \text{Loss}(f)$

- $\text{Loss}(f) = \text{EmpiricalLoss}(f) + \text{GeneralizeLoss}(f)$

Regression

- Find a function, f, to minimize
  error on examples,
  and generalization (previously unseen) error

- $f = \min_f \text{Loss}(f)$

- $\text{Loss}(f) = \text{EmpiricalLoss}(f) + \text{GeneralizeLoss}(f)$

- $\text{EmpiricalLoss}(f) = \sum_i (y_i - f(x_i))^2$

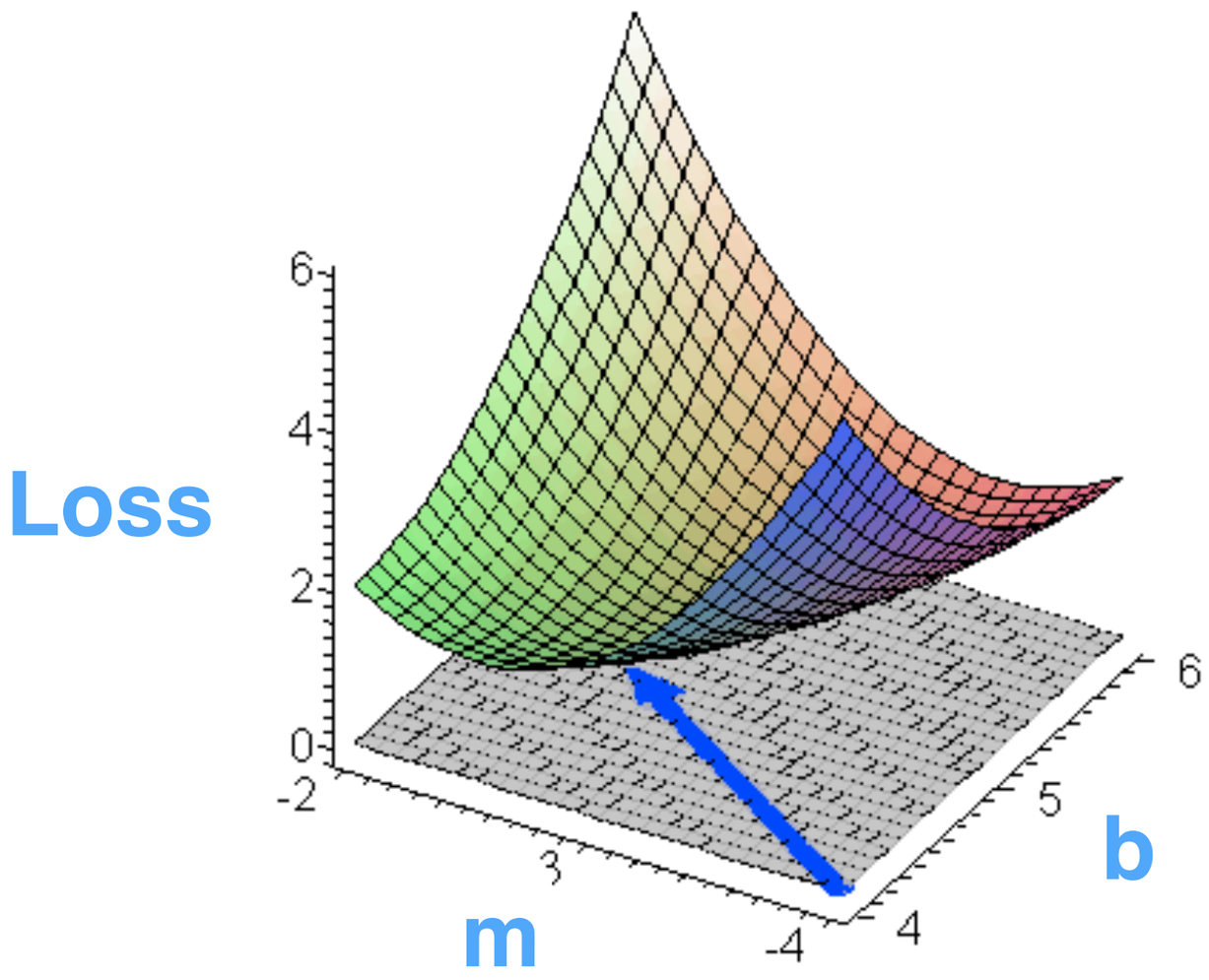
Regression

- Find a function, f, to minimize
  error on examples,
  and generalization (previously unseen) error

- $f = \min_f \text{Loss}(f)$
- $\text{Loss}(f) = \text{EmpiricalLoss}(f) + \text{GeneralizeLoss}(f)$
- $\text{EmpiricalLoss}(f) = \sum_i (y_i - f(x_i))^2$
- Linear Regression: calculate exact answer (Gauss)

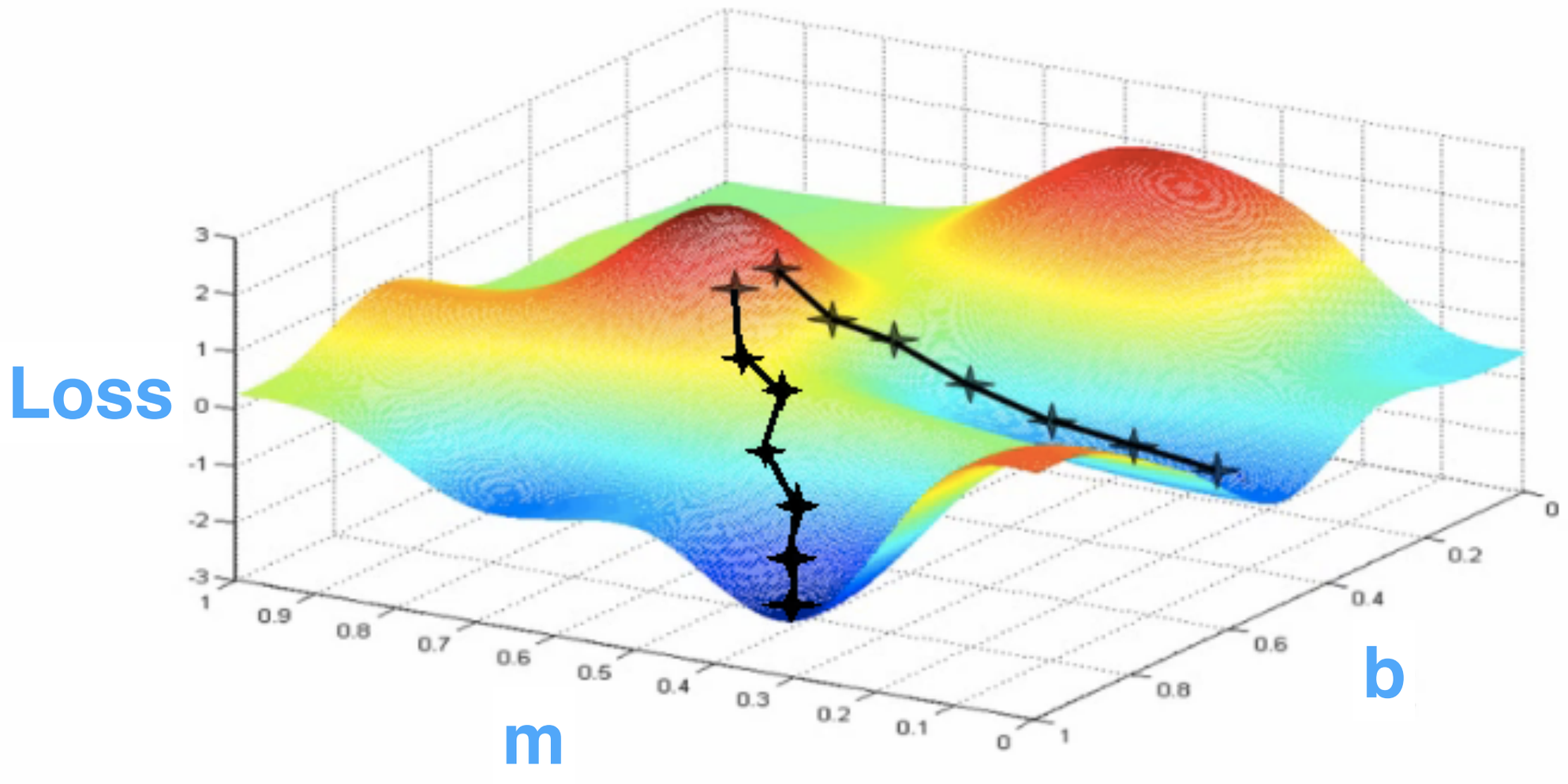
Gradient Descent

$$f(x) = m\,x + b$$

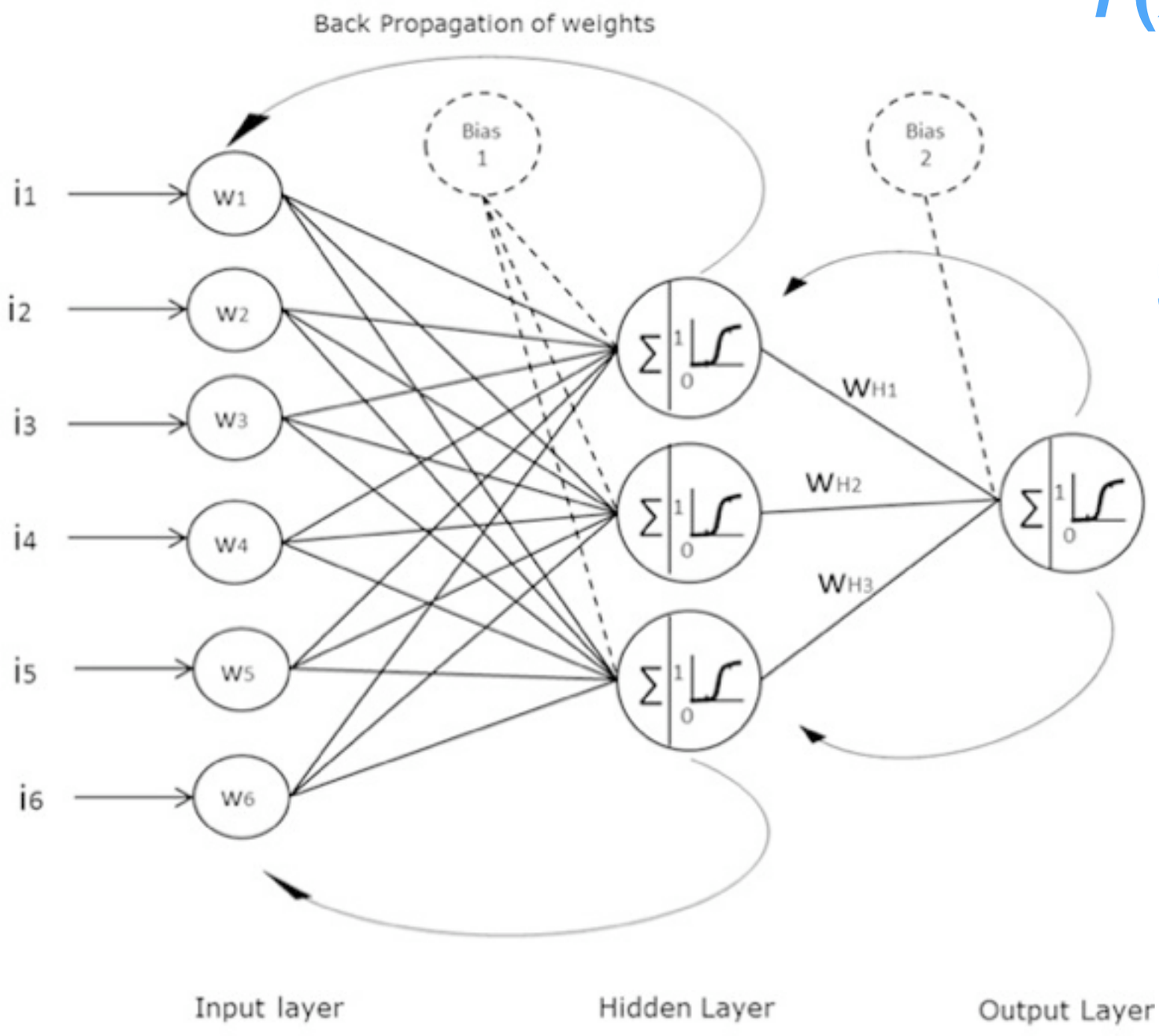


Gradient Descent

f (x) = nonlinear function of x



Neural Network

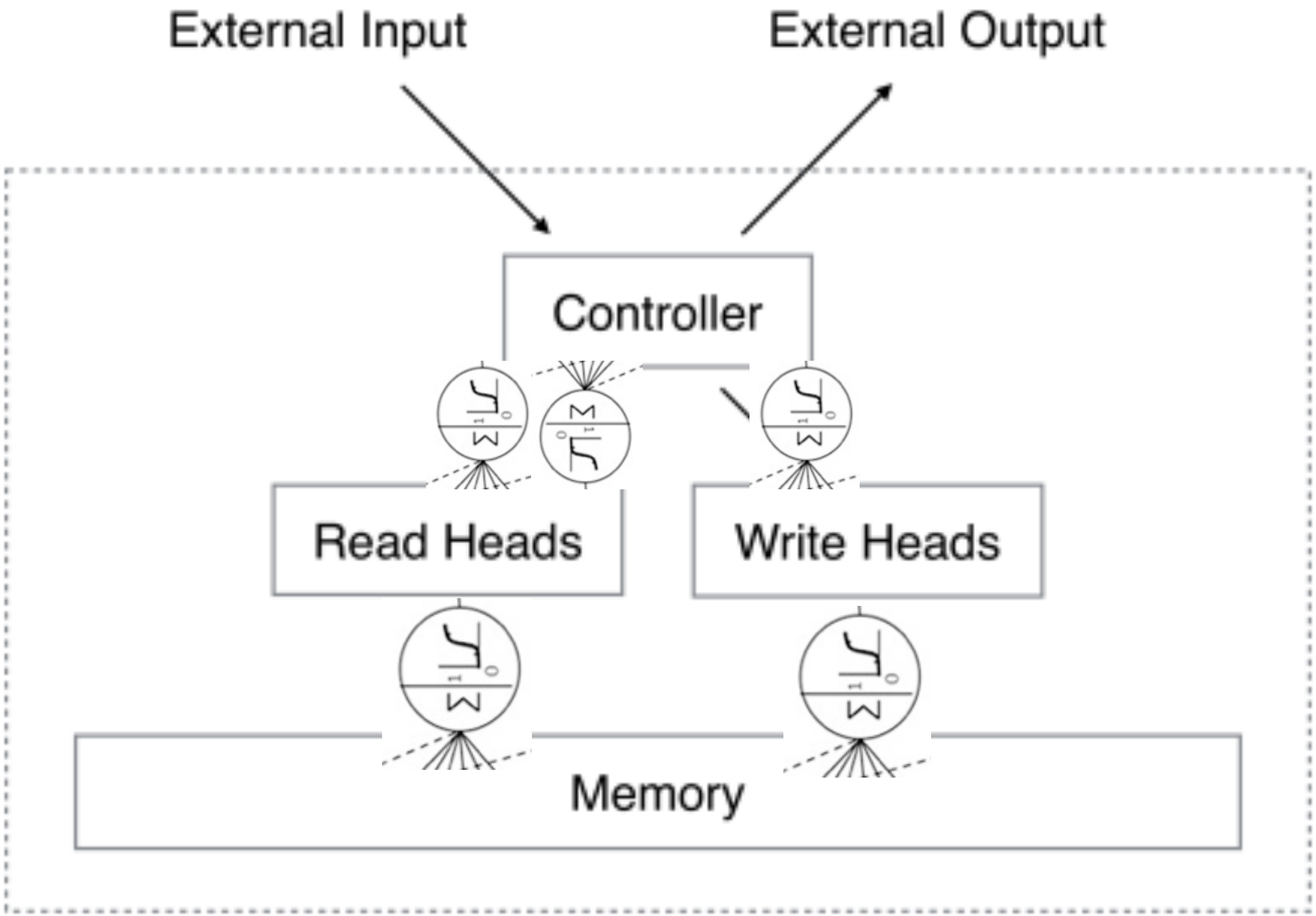


f (X) = nonlinear function composed of

Σ, Π, S

S(x) = 1 / (1 - e^x)

Neural Turing Machines



Learns programs like “copy input array to output”

Q: Can we learn **complex nontraditional programs** from examples?

A: **Not yet**,
maybe someday.

Q: Can we learn to **optimize** programs?

When in doubt,
use brute force
— Ken Thompson

Superoptimization

- Henry Massalin, 1987
  Given a short function f, try all sequences a few assembler instructions long

- Alex Aiken 2006
  Slightly longer f; more clever and less brute force

Q: Can we learn to **optimize** programs?

A: **Yes,** short parts.

Q: Can we learn to **efficiently execute declarative programs**?

Program Optimization: Crossing the Math/CS Chasm

$$\begin{aligned}\mathbf{e}_{\text{best}} &= \text{argmax}_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) \\ &= \text{argmax}_{\mathbf{e}} p(\mathbf{f}|\mathbf{e}) p_{\text{LM}}(\mathbf{e}) \omega^{\text{length}(\mathbf{e})}\end{aligned}$$

- A few lines of math
- Tens of thousands of lines of C++

Package	Files	SLOC	Language	Application area
SRILM	285	48967	C++	Language modeling
Charniak parser	266	42464	C++	Parsing
Stanford parser	417	134824	Java	Parsing
cdec	178	21265	C++	Machine translation
Joshua	486	68160	Java	Machine translation
MOSES	351	37703	C++	Machine translation
GIZA++	122	15958	C++	Bilingual alignment
OpenFST	157	20135	C++	Weighted FSAs & FSTs
NLTK	200	46256	Python	NLP education
HTK	111	81596	C	Speech recognition
MALLET	620	77155	Java	Conditional Random Fields
GRMM	90	12926	Java	Graphical model add-on
Factorie	164	12139	Scala	Graphical models

Declarative Languages for AI, ML Applications

- DYNA (Jason Eisner)
  Probabilistic Context Free Parser:

```
% A single word is a phrase (given an appropriate grammar rule).
phrase(X,I,J) += rewrite(X,W) * word(W,I,J).
% Two adjacent phrases make a wider phrase (given an appropriate rule).
phrase(X,I,J) += rewrite(X,Y,Z) * phrase(Y,I,Mid) * phrase(Z,Mid,J).
% An phrase of the appropriate type covering the whole sentence is a parse.
goal          += phrase(start_nonterminal,0,length).
```

```
rewrite("S","NP","VP")=0.9

word("spring",5,6)=1

start_nonterminal="S"
```

Declarative Languages for AI, ML Applications

- DYNA (Jason Eisner)
  Probabilistic Context Free Parser:

```
% A single word is a phrase (given an appropriate grammar rule).
phrase(X,I,J) += rewrite(X,W) * word(W,I,J).
% Two adjacent phrases make a wider phrase (given an appropriate rule).
phrase(X,I,J) += rewrite(X,Y,Z) * phrase(Y,I,Mid) * phrase(Z,Mid,J).
% An phrase of the appropriate type covering the whole sentence is a parse.
goal          += phrase(start_nonterminal,0,length).
```

```
rewrite("S","NP","VP")=0.9
word("spring",5,6)=1
start_nonterminal="S"
```

Probabilistic / Relational Languages: any variable can be input or output

Q: Can we learn to
efficiently execute declarative programs?

A: **Maybe.** So far,
not done via learning.
Need help with languages.

Q: Can we learn an **interpreter**?

Recurrent Neural Net Learning with LSTM

Input:

```
f=(8794 if 8887<9713 else (3*8334))
print((f+574))
```

Target: 9368.
Model prediction: 9368.

Input:

```
j=8584
for x in range(8):
  j+=920
b=(1500+j)
print((b+7567))
```

Target: 25011.
Model prediction: 23011.

Input:

```
c=445
d=(c-4223)
for x in range(1):
  d+=5272
print((8942 if d<3749 else 2951))
```

Target: 8942.
Model prediction: 8942.

Zaremba & Sutskever:
Learning to Execute

Q: Can we learn an **interpreter**?

A: **Partly**, but that's not the point.

Q: Can we learn a **user's language**?

WolframAlpha computational... knowledge engine

Enter what you want to calculate or know about:

x^1/2

Examples

Input:

$$\sqrt{x}$$

WolframAlpha computational knowledge engine

x^2/2

Examples

Input:

$$\frac{x^2}{2}$$

Q: Can we learn a **user's language**?

A: **Interesting idea,** but so far done by hand.

Q: Can we learn
tutoring feedback
from examples?

```
if x = 1:
     ^
SyntaxError: invalid syntax
```

```
if x == 0
    print(‘x is zero')
```

```
if x == 0
          ^
SyntaxError: invalid syntax
```

```
    print('x is zero')
    ^
IndentationError: unexpected indent
```

```
def qsort(A):
    if len(A) <= 1:
        return A
    LT, EQ, GT = []
    pivot = A[0]
    for x in A:
        if x < pivot:    LT.append(x)
        elif x > pivot: GT.append(x)
        else:                EQ.append(x)
    return qsort(LT) + EQ + qsort(GT)
```

```
def qsort(A):
    if len(A) <= 1:
        return A
    LT = EQ = GT = []
    pivot = A[0]
    for x in A:
        if x < pivot:    LT.append(x)
        elif x > pivot: GT.append(x)
        else:                EQ.append(x)
    return qsort(LT) + EQ + qsort(GT)
```

```
def qsort(A):
    if len(A) <= 1:
        return A
    LT, EQ, GT = [], [], []
    pivot = A[0]
    for x in A:
        if x < pivot:   LT.append(x)
        elif x > pivot: GT.append(x)
        else:           EQ.append(x)
    return qsort(LT) + EQ + qsort(GT)
```

python eror correctoin

Web Images Maps Shopping More Search tools

About 181,000 results (0.60 second…

Showing results for python ***error correction***

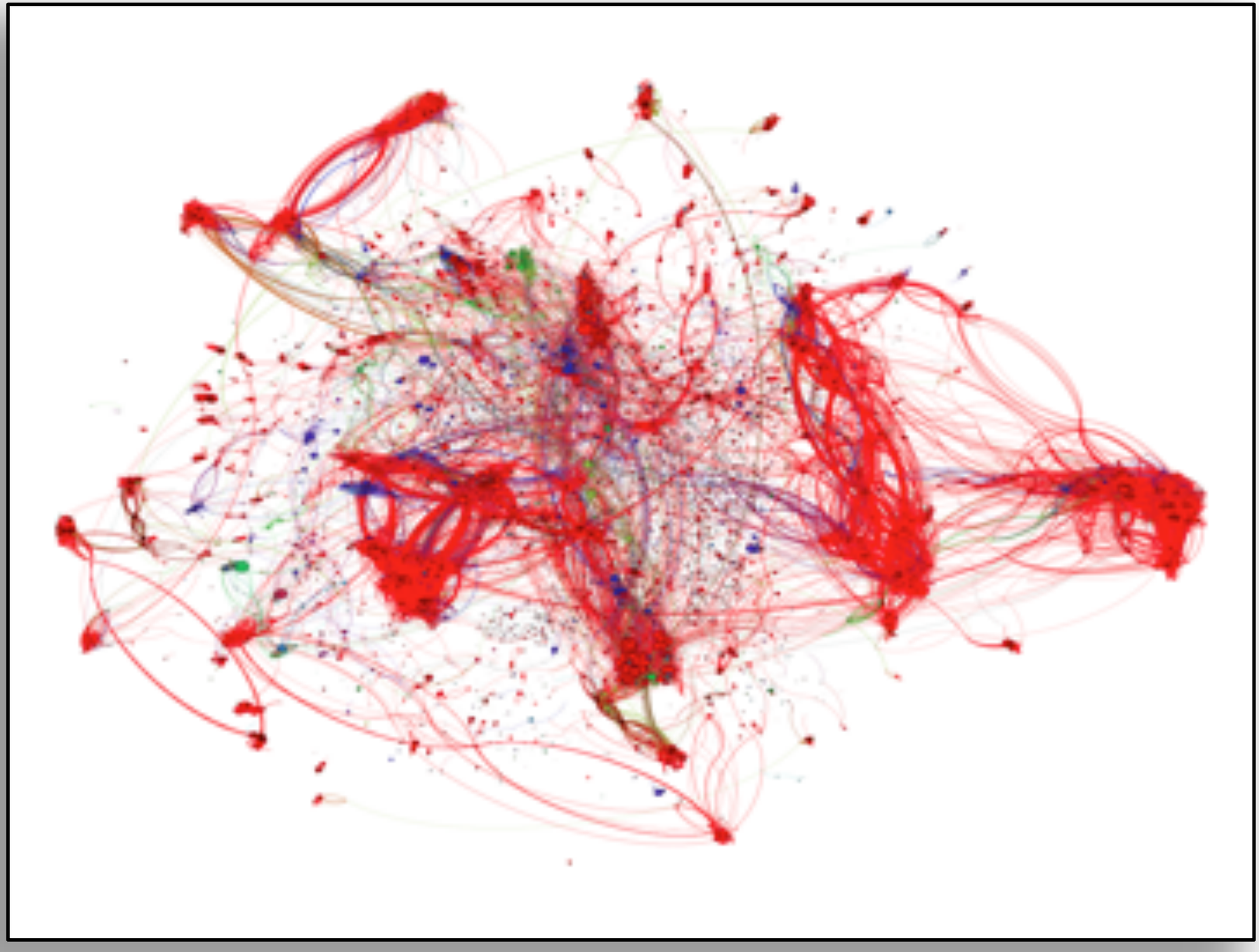
Search instead for python eror correctoin

error correction - Reed-Solomon Decoding - Stack Overflow

stackoverflow.com/questions/1672447/reed-solomon-decoding

1 answer - Nov 4, 2009

I've tried the **Python** ReedSolomon module, but I'm not even sure how to configure ...
When is forward **error correction** a good idea for packets?



Piech & Huang

Group together similar errors; Give advice for all in class

```
sum(transpose(X*theta-y)*X)

sum(X'*(X*theta-y))

sum(((theta'*X')'-y)'*X)
```

Force Multiplication Results using learned program embeddings

After having an expert grade 500:

Algorithm	Submissions Covered	True Positive Rate	Force Multiplication
None	500	100%	1.0
Logistic Regression	35,052	98.7%	70.1
Kmeans Active Learning + Logistic Regression	47,397	98.8%	**94.8**

Feedback for all 71,000 unique programs was generated automatically from a complex script. Feedback was chosen from 14 different discrete labels.

[piech et al]

Q: Can we learn **tutoring feedback** from examples?

A: **Yes!** Works better as a human/machine partnership.

Discussion