

CS6120

Kenneth Church

<https://kwchurch.github.io>

Kenneth Church: Home Page



Northeastern University

1. [Computer Science](#)
2. [The Institute for Experiential AI \(EAI\)](#)

I work in computational linguistics and related areas such as: web search, language modeling, text analysis, spelling correction, word-sense disambiguation, terminology, translation, lexicography, compression, optical character recognition, speech (recognition, synthesis, and diarization), and more. I was an early advocate of empirical methods and a founder of the Conference on Empirical Methods in Natural Language Processing (EMNLP).

Kenneth was the president of the Association for Computational Linguistics (ACL) in 2012 and SIGDAT (the group that organizes EMNLP) from 1993 until 2011. He became an AT&T Fellow in 2001 and ACL Fellow in 2015.

Teaching

[2023-Fall: CS 6120](#)

<https://kwchurch.github.io>

CS6120: Practical Natural Language Processing

[Kenneth Church](#)

Previous versions of this class

1. [Spring 2023 in San Jose](#)
2. [Spring 2023 in Boston](#)

Textbooks

1. [JM3](#)
2. [E](#) (optional; maybe)
3. [Deep Learning with Python](#) (optional; maybe)

Computers

1. Your laptop; will need to install python (and more)
2. Koury Cloud: [this](#) and [this](#)
3. [Discovery Cluster](#)
4. [GitHub](#) (please request an account)

The syllabus below is modeled after the previous versions of this class, though I hope to prioritize the material based more on current priorities and less on how the field got to where it is.

Syllabus

Date	Topic	Teacher	Slides	Readings	Assignments
9/11	Introduction, Tools, Machine Learning, Statistics	Kenneth Church	Slides	<ol style="list-style-type: none">1. JM3: Chapter 62. My opinion piece on Word2vec3. Church and Hanks (1990)4. Latent Semantic Indexing	Better Together (Due 9/23)
9/18	Bag of Words, tf/idf, Naive Bayes, PMI, Word2vec, Man is to Woman as..., Linear Algebra, Rotations, Bilingual Lexicon Induction (BLI), BERT	Jiaji Huang	Slides Lecture from Last Term	<ol style="list-style-type: none">1. General Fine-Tuning2. HuggingFace tutorial on inference	No Assignment
9/25	Deep Nets: Inference	Kenneth Church	Slides	JM3: Chapter 14	Assignment 2 (Due 10/14)
10/2	More Deep Nets: Question Answering, Machine Translation, Part of Speech Tagging, Sequence Modeling	Kenneth Church	Slides	No Readings (Study for Exam)	No Assignment (Study for Exam)
10/9	Holiday				

Goals for this class

- Prioritize material based
 - more on current priorities and
 - less on how
 - the field got to where it is
- Start with what's hot now
 - Deep nets and Bots
- End with
 - Topics that may become great (again)
- Success:
 - Students have confidence that
 - they can learn
 - what you need to know
 - just in time
 - (when you need to know it)
 - Learn to swim by
 - jumping into deep end
 - Won't let you drown
 - (but no spoon feeding)

Grading, Accessibility, Tardiness, Effort, etc.

- Final Project:
 - Encourage Team Efforts
 - Proposal & Final Presentation
 - Coding
 - Written Presentation
 - Oral Presentation
- Mid-Term Exam
 - Open book, computer, network
 - If you can find a way to use ChatGPT to do the exam for you
 - Shame on me!
- Flipped Classroom
 - Lectures → Recitations
 - Homework → Readings/Lectures
 - Assignments → Exercises
- First Assignment → Calibration
 - Expect wide range of backgrounds
 - Don't be discouraged if others find it too easy (and you don't)
 - I want to see effort/progress
 - If they already know it
 - They can't learn it
- Readings
 - Intended to prepare for next class

Better Together Homework

- Purpose(s)
 - Motivate
 - What's hot: deep nets, embeddings
 - and what's not: python, pip, etc.
 - Calibration
 - I'm clueless about...
 - Avoid getting into details
 - What is an embedding?
 - What is a deep net?
 - Little Languages
 - Suggest that deep nets are like regression

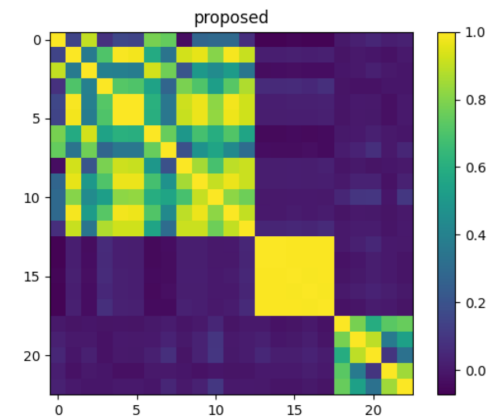
Better Together

This assignment is based on [this](#).

Please try [this](#) and [this](#). Many names such as *David Madigan* and *Noah Smith* are difficult for search engines such as [Semantic Scholar](#). This exercise provides a visualization to help resolve some of these ambiguities.

Tasks

Write a program that inputs a name such as *Noah Smith* and outputs a visualization like this:



Note: Your picture will look different because you will be using different embeddings.

Please post code and output pictures on GitHub, and share links to your code on Canvas.

Suggested steps:

1. [input name and output list of candidates and their papers](#)
2. [input paper id and output embeddings](#)
3. [compute pairwise similarities](#)
4. [plot similarities](#)

Little Languages

Little Languages (from Unix)

In the good old days, real programmers would swagger to a key punch and, standing the whole time, crank out nine cards like:

```
//SUMMARY JOB REGION=( 100K,50K )
// EXEC PGM=SUMMAR
//SYSIN DD DSNAME=REP.8601,DISP=OLD,
// UNIT=2314,SPACE=(TRK,(1,1,1)),
// VOLUME=SER=577632
//SYSOUT DD DSNAME=SUM.8601,DISP=(,KEEP),
// UNIT=2314,SPACE=(TRK,(1,1,1)),
// VOLUME=SER=577632
//SYSABEND DD SYSOUT=A
```

Today's young whippersnappers do this simple job by typing

```
summarize <jan.report >jan.summary
```

programming pearls

by Jon Bentley

LITTLE LANGUAGES

When you say "language," most programmers think of the big ones, like FORTRAN or COBOL or Pascal. In fact, a language is any mechanism to express intent, and the input to many programs can be viewed profitably as statements in a language. This column is about those "little languages."

Programmers deal with microscopic languages every day. Consider printing a floating-point number in six characters, including a decimal point and two subsequent digits. Rather than writing a subroutine for the task, a FORTRAN programmer specifies the format F6.2, and a COBOL programmer defines the picture 999.99. Each of these descriptions is a statement in a well-defined little language. While the languages are quite different, each is appropriate for its problem domain: although a FORTRAN programmer might complain that 999999.99999 is too long when F12.5 could do the job, the COBOLer can't even express in FORTRAN such common financial patterns as \$, \$\$\$, \$\$9.99. FORTRAN is aimed at scientific computing, COBOL is designed for business.

In the good old days, real programmers would swagger to a key punch and, standing the whole time, crank out nine cards like:

```
//SUMMARY JOB REGION=( 100K,50K )
// EXEC PGM=SUMMAR
//SYSIN DD DSNAME=REP.8601,DISP=OLD,
// UNIT=2314,SPACE=(TRK,(1,1,1)),
// VOLUME=SER=577632
//SYSOUT DD DSNAME=SUM.8601,DISP=(,KEEP),
// UNIT=2314,SPACE=(TRK,(1,1,1)),
// VOLUME=SER=577632
//SYSABEND DD SYSOUT=A
```

Today's young whippersnappers do this simple job by typing

```
summarize <jan.report >jan.summary
```

Modern successors to the old "job control" languages are not only more convenient to use, they are fundamentally more powerful than their predecessors. In

the June column, for instance, Doug McIlroy implemented a program to find the K most common words in a document in six lines of the UNIX® SHELL language.

Languages surround programmers, yet many programmers don't exploit linguistic insights. Examining programs under a linguistic light can give you a better understanding of the tools you now use, and can teach you design principles for building elegant interfaces to your future programs. This column will show how the user interfaces to half a dozen interesting programs can be viewed as little languages.

This column is built around Brian Kernighan's PIC language for making line drawings. Its compiler is implemented on the UNIX system, which is particularly supportive (and exploitative) of language processing; the sidebar on pages 714-715 shows how little languages can be implemented in a more primitive computing environment (BASIC on a personal computer).

The next section introduces PIC and the following section compares it to alternative systems. Subsequent sections discuss little languages that compile into PIC and little languages used to build PIC.

The PIC Language

If you're talking about compilers, you might want to depict their behavior with a picture:

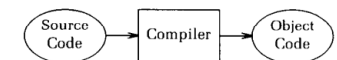
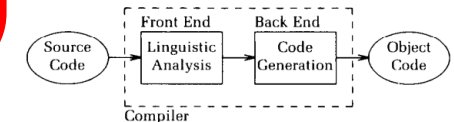


FIGURE 1. A Simple View of a Compiler

(This diagram is genuine PIC output; we'll see its input description shortly.) Or you may desire a little more detail about the internal structure:



UNIX is a trademark of AT&T Bell Laboratories.

We propose *gft*, a Little Language

In the good old days, real programmers would swagger to a key punch and, standing the whole time, crank out nine cards like:

```
//SUMMARY  JOB  REGION=( 100K,50K )
//          EXEC PGM=SUMMAR
//SYSIN     DD  DSNAME=REP.8601,DISP=OLD,
//          UNIT=2314,SPACE=(TRK,(1,1,1)),
//          VOLUME=SER=577632
//SYSOUT    DD  DSNAME=SUM.8601,DISP=(,KEEP),
//          UNIT=2314,SPACE=(TRK,(1,1,1)),
//          VOLUME=SER=577632
//SYSABEND DD  SYSOUT=A
```

Today's young whippersnappers do this simple job by typing

```
summarize <jan.report >jan.summary
```

- In the good old days,
 - real programmers would copy a few hundred lines of
 - PyTorch from HuggingFace
 - and modify as necessary to change
 - models,
 - datasets,
 - task,
 - etc.
- With *gft*, 100s of lines → 1-line
 - More accessible to masses
 - (including non-programmers)

Standard 3-Step Recipe

<u>Step</u>	<u>Description</u>	<u>Time</u>	<u>Hardware</u>
1			
2			
3	Inference (predict)	Seconds/Minutes	0+ GPUs



Easy

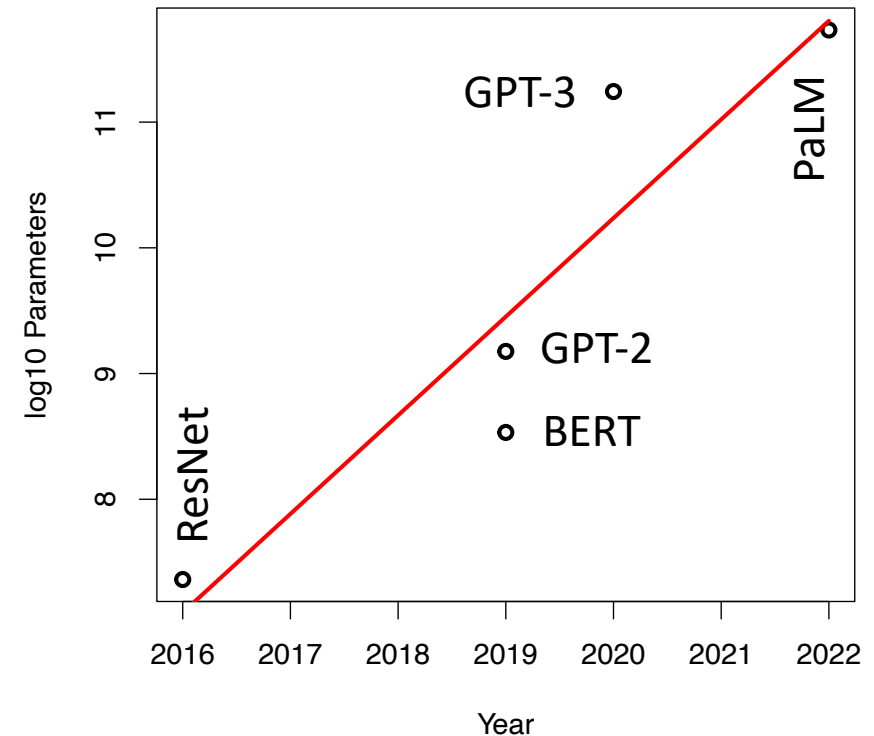
Standard 3-Step Recipe

Most users
should not do
this themselves

<u>Step</u>	<u>Description</u>	<u>Time</u>	<u>Hardware</u>
1	Pre-Training	Days/Weeks	Large GPU cluster
2			
3	Inference (predict)	Seconds/Minutes	0+ GPUs


Hard: Pre-Training Large Language Models (LLMs)

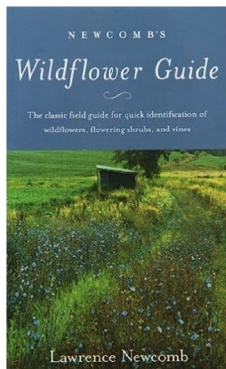
Year	Deep nets	Billions of parameters
2016	ResNet-50 (He <i>et al.</i> , 2016)	0.023
2019	BERT (Devlin <i>et al.</i> , 2019)	0.34
2019	GPT-2 (Radford <i>et al.</i> , 2019)	1.5
2020	GPT-3 (Brown <i>et al.</i> , 2020; Dale 2021)	175
2022	PaLM (Chowdhery <i>et al.</i> , 2022)	540



Most users should not invest in pretraining because growth (& costs) are out of control

Standard 3-Step Recipe

<u>Step</u>	<u>Description</u>	<u>Time</u>	<u>Hardware</u>
1	Pre-Training	Days/Weeks	Large GPU cluster
2	 Fine-Tuning (fit)	Hours/Days	1+ GPUs
3	Inference (predict)	Seconds/Minutes	0+ GPUs



Example of Fine-Tuning (aka, *fit*)

fit: $f_{pre} + data \rightarrow f_{post}$

f_{pre} : Resnet

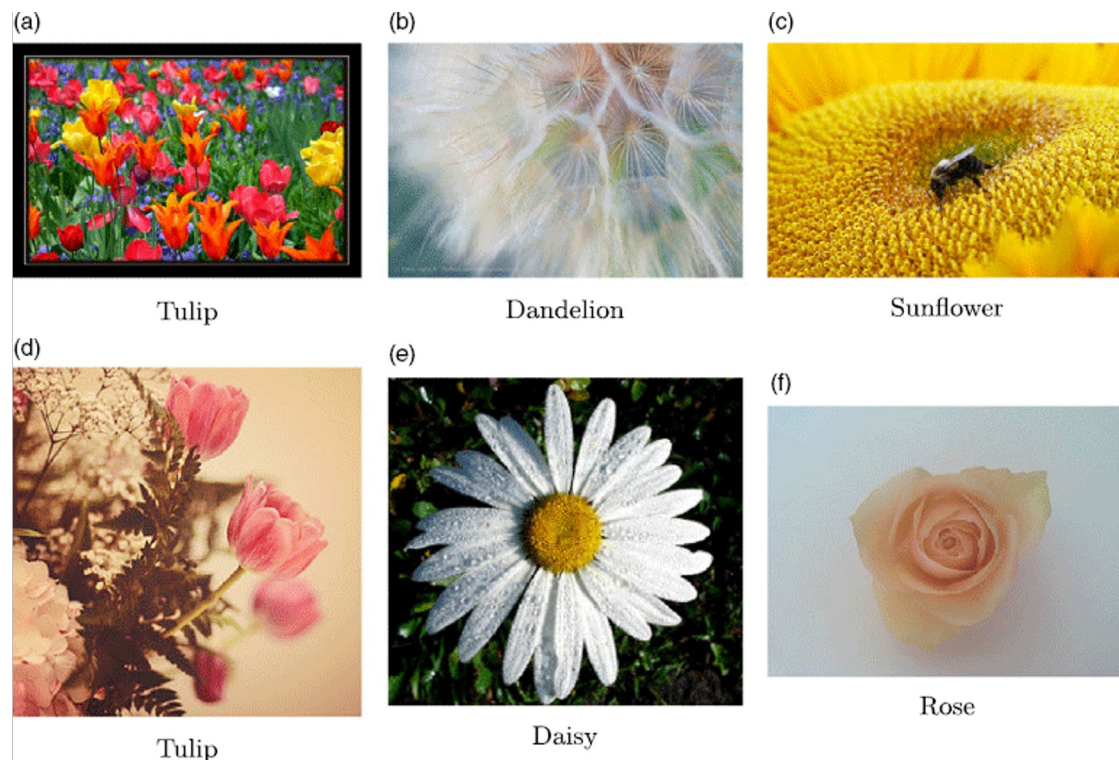
- Maps images (jpg files) \rightarrow classes (strings)
- Trained on ImageNet
 - input (x): 14M images (of many things)
 - output (y): 1000 classes (strings)

- **data : flowers**

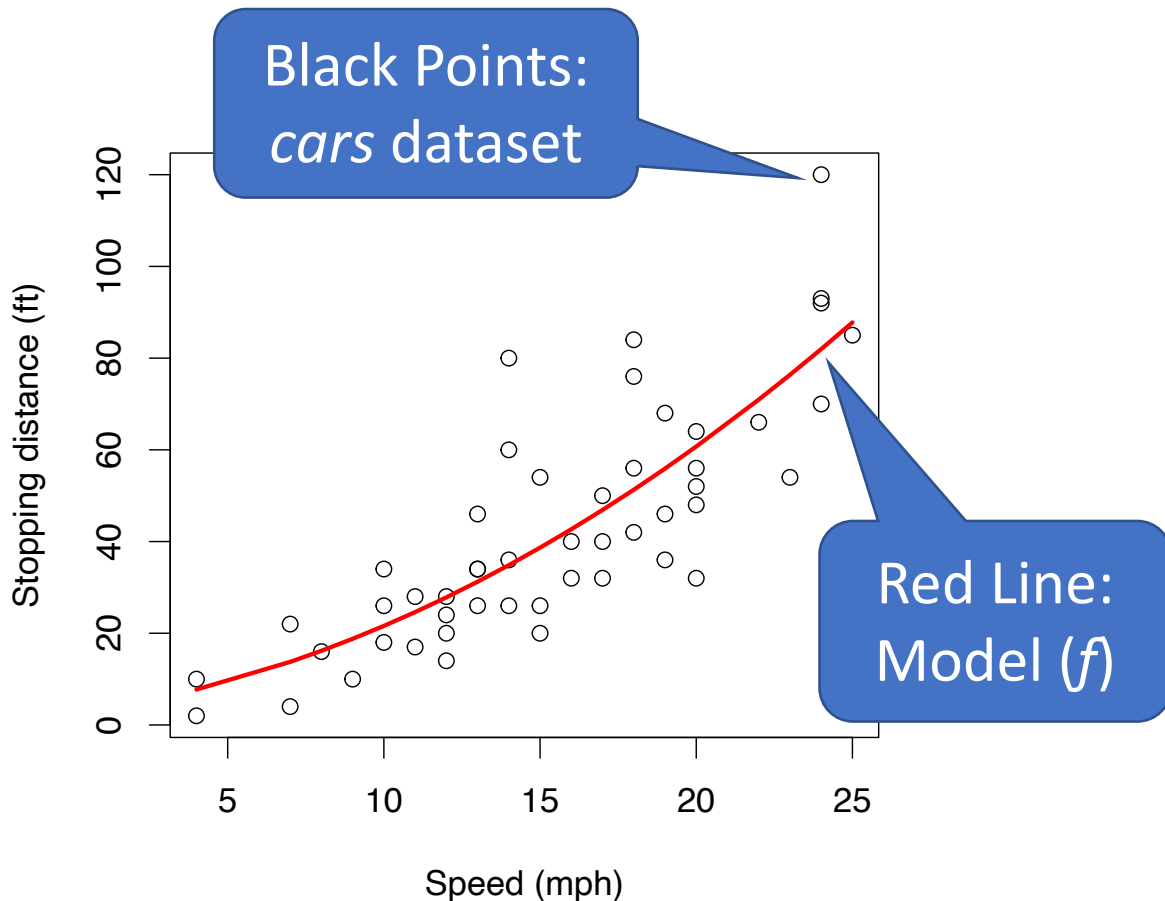
- input (x): 2195 pictures of flowers
- outputs (y): 5 classes of flowers
 - *Tulip, Dandelion, Sunflower, Daisy, Rose*

- f_{post} :

- Maps images (jpg files) \rightarrow flowers (strings)
- Reject modeling is hard



Fine-Tuning (Fit) in R (Statistics Package)

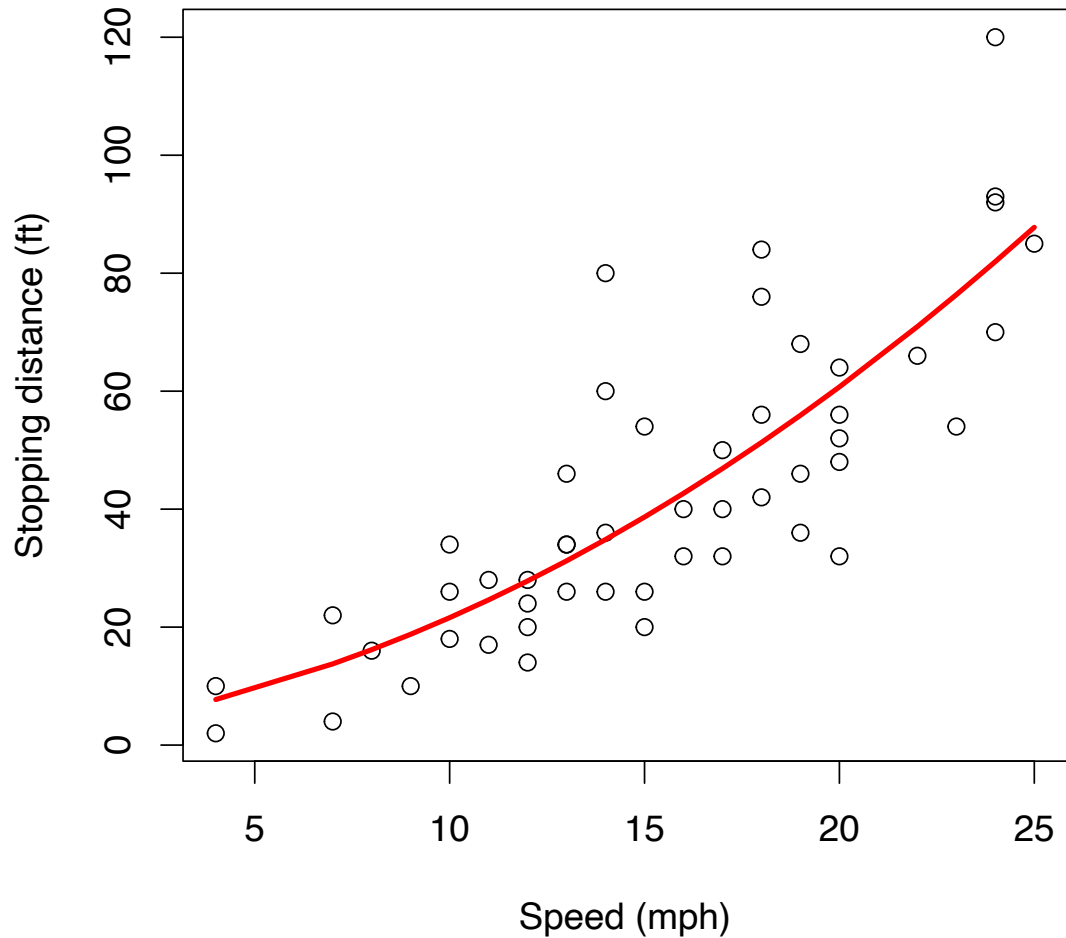


Make deep nets look like regression
(Not much programming)

- Notational Conventions:
 - Observations: Circles
 - Models (f): Red Lines
- Prediction: $f(x)$
 - Use model (f) to map
 - input x (speed) to
 - output y (stopping distance)
 - For linear regression,
 - f is a polynomial
 - For gft,
 - f is typically a model from a hub

Datasets

Example: Cars



```
> head(cars)
  speed  dist
1     4     2
2     4    10
3     7     4
4     7    22
5     8    16
6     9    10
```

Two Columns:
1. cars\$speed
2. cars\$dist

Example of Datasets in HuggingFace

<https://huggingface.co/dair-ai/emotion>

 Hugging Face

Search models, datasets, users...

Models Datasets Spaces Docs

Datasets: [dair-ai/emotion](#) like 114

Tasks: [Text Classification](#) Sub-tasks: [multi-class-classification](#) Languages: [English](#) Multilinguality: [monolingual](#) Size Categories: [10K<n<100K](#) Language Creation: [other](#)

Annotations Creators: [machine-generated](#) Source Datasets: [original](#) Tags: [emotion-classification](#) License: [other](#)

Dataset card Files and versions Community 6

```
> head(cars)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

Dataset Viewer

Auto-converted to Parquet [API](#) [Go to dataset viewer](#)

Subset

split (20k rows)

Split

train (16k rows)

Search this dataset

text (string)

label (class label)

"i didnt feel humiliated" 0 (sadness)

"i can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake" 0 (sadness)

"im grabbing a minute to post i feel greedy wrong" 3 (anger)

"i am ever feeling nostalgic about the fireplace i will know that it is still on the property" 2 (love)

"i am feeling grouchy" 3 (anger)

"ive been feeling a little burdened lately wasnt sure why that was" 0 (sadness)

"ive been taking or milligrams or times recommended amount and ive fallen asleep a lot faster but i also feel like so funny" 5 (surprise)

Downloads last month 7,358

[Use in dataset library](#) [Edit dataset card](#)

[Train in AutoTrain](#) [Papers with Code](#)

[Evaluate models](#) [HF Leaderboard](#)

Homepage: [github.com](#) Size of downloaded dataset files: 16.1 MB

Size of the auto-converted Parquet files: 28.2 MB Number of rows: 436,809

Models trained or fine-tuned on dair-ai/emotion

[aiknowyou/it-emotion-analyzer](#)

Emotion → GLUE (Cola)

<https://huggingface.co/datasets/glue/viewer/cola/train>

```
> head(cars)
  speed  dist
1     4     2
2     4    10
3     7     4
4     7    22
5     8    16
9     9    10
```

Hugging Face Search models, datasets, users... Models Datasets Spaces Docs Solutions Pricing

Datasets: glue like 29

Tasks: natural-language-inference acceptability-classification text-classification-other-paraphrase-identification +4 Task Categories: text-classification text-scoring Languages: en Multilingual

Size Categories: 10K<n<100K Licenses: cc-by-4-0 Language Creators: unknown Annotations Creators: unknown Source Datasets: unknown

Dataset card Files and versions

Terminology:
Subset

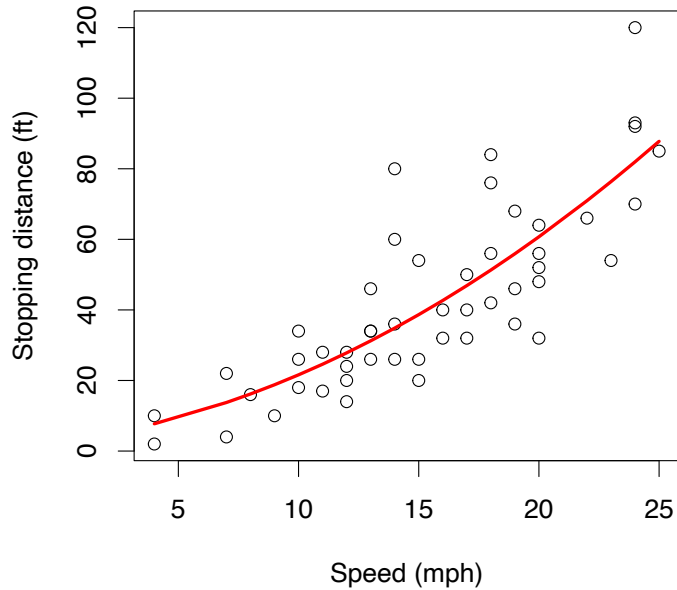
Terminology:
Splits

Dataset Preview

Subset: cola

sentence (string)	label (class label)	idx (int)
Our friends won't buy this analysis, let alone the next one we propose.	1 (acceptable)	0
One more pseudo generalization and I'm giving up.	1 (acceptable)	1
One more pseudo generalization or I'm giving up.	1 (acceptable)	2
The more we study verbs, the crazier they get.	1 (acceptable)	3
Day by day the facts are getting murkier.	1 (acceptable)	4
I'll fix you a drink.	1 (acceptable)	5
Fred watered the plants flat.	1 (acceptable)	6
Bill coughed his way out of the restaurant.	1 (acceptable)	7
We're dancing the night away.	1 (acceptable)	8
Herzhan hammered the metal flat.	1 (acceptable)	9
The critics laughed the play off the stage.	1 (acceptable)	10

glm (General Linear Models) in R (and Sklearn)



```
3 # Create the black points
4 plot(cars, xlab="Speed (mph)", ylab="Stopping distance (ft)")
7 g = glm(dist ~ poly(speed, 2), data=cars)
10 o = order(cars$speed)
11 # Show predictions as a red line
12 lines(cars$speed[o], predict(g,cars)[o], col="red", lwd=3)
```

glm: fit poly model
with data

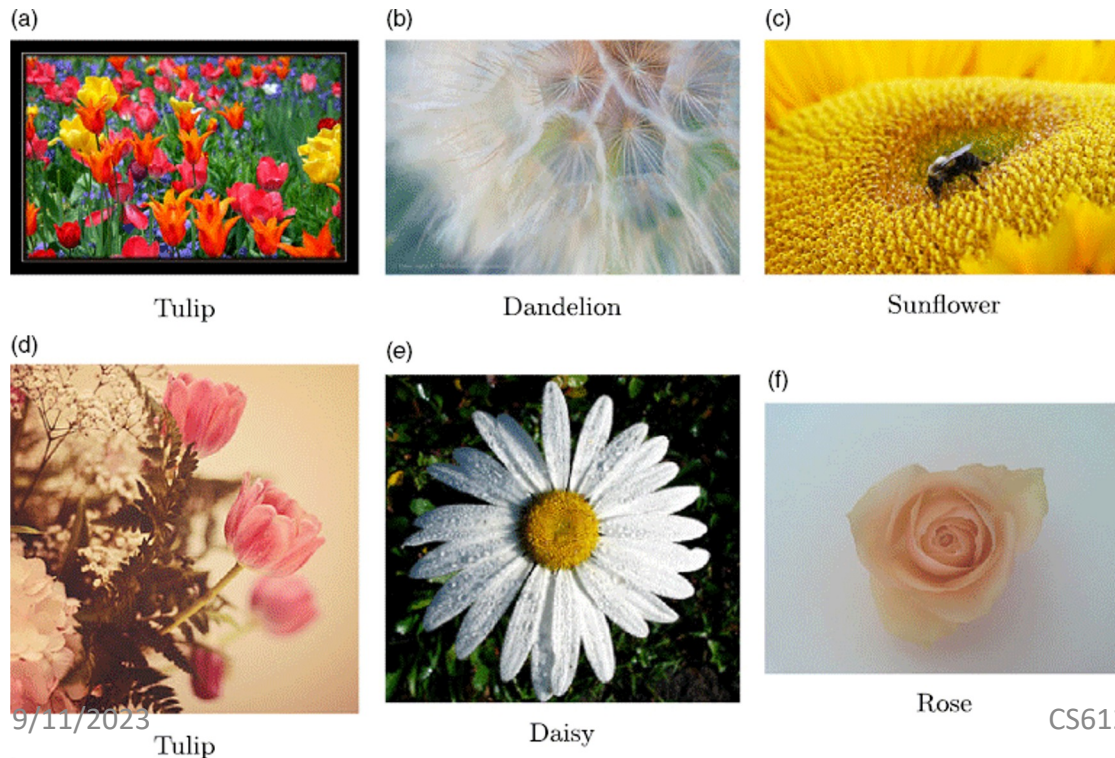
predict: $dist \approx g(cars\$speed)$

Fine-Tuning (fit) in GFT

$f_{pre} + data \rightarrow f_{post}$

- Flowers

- f_{pre} : Resnet
- $data$: flowers



```
gft_fit --eqn 'classify: label ~ text' \  
--model H:bert-base-cased \  
--data H:emotion \  
--output_dir $outdir
```

f_{pre} : Pre-trained Model

f_{post} : Post-trained Model

```
7 g = glm(dist ~ poly(speed, 2), data=cars)
```

Fine-Tuning (fit) in R



Datasets: emotion like 18

Tasks: multi-class-classification text-classification-other-emotion-classification Task Categories: text-classification

Language Creators: machine-generated Annotations Creators: machine-generated Source Datasets: original

Dataset card Files and versions

Dataset Preview

Subset

default

Split

train

text (string)

label (class label)

i didnt feel humiliated

0 (sadness)

i can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake

0 (sadness)

im grabbing a minute to post i feel greedy wrong

3 (anger)

i am ever feeling nostalgic about the fireplace i will know that it is still on the property

2 (love)

i am feeling grouchy

3 (anger)

ive been feeling a little burdened lately wasnt sure why that was

0 (sadness)

ive been taking or milligrams or times recommended amount and ive fallen asleep a lot faster but i also feel like so funny

5 (surprise)

i feel as confused about life as a teenager or as jaded as a year old man

4 (fear)

i have been with petronas for years i feel that petronas has performed well and made a huge profit

1 (joy)

i feel romantic too

2 (love)

i feel like i have to make the suffering i m seeing mean something

0 (sadness)

Emotion Dataset

classify: label~text

Task/Loss

Columns

Row

Fine-Tuning (fit): Numerous Use Cases

$$f_{pre} + data \rightarrow f_{post}$$

- Flowers
 - f_{pre} : Resnet
 - $data$: flowers
- Emotion Classification
 - f_{pre} : <https://huggingface.co/bert-base-uncase>
 - $data$: <https://huggingface.co/datasets/dair-ai/emotion>
- GLUE
- SQuAD
- Machine Translation
- Speech Recognition
- Vision
- and much more

```
gft_fit --eqn 'classify: label ~ text' \  
--model H:bert-base-cased \  
--data H:emotion \  
--output_dir $outdir
```

f_{pre} : Pre-trained Model

f_{post} : Post-trained Model

```
7 g = glm(dist ~ poly(speed, 2), data=cars)
```

Fine-Tuning (fit) in R

GLUE Subsets

Subset	Dataset
COLA	H:glue,cola
SST2	H:glue,sst2
WNLI	H:glue,wnli
MRPC	H:glue,mrpc
QNLI	H:glue,qnli
QQP	H:glue,qqp
SSTB	H:glue,sstb
MNLI	H:glue,mnli

GLUE COLA SUBSET	
Sentence	Label
Bill sang himself to sleep.	1 (acceptable)
Bill squeezed the puppet through the hole.	1 (acceptable)
Bill sang Sue to sleep.	1 (acceptable)
The elevator rumbled itself to the ground.	0 (unacceptable)
If the telephone rang, it could ring itself silly.	1 (acceptable)
She yelled hoarse.	0 (unacceptable)
Ted cried to sleep.	0 (unacceptable)
The tiger bled to death.	1 (acceptable)

GFT Program for COLA

https://github.com/kwchurch/gft/blob/master/examples/fit_examples/model.HuggingFace/language/data.HuggingFace/glue/cola.sh

```
1  #!/bin/sh
2
3  echo hostname = `hostname`
4
5  gft_fit --model H:bert-base-cased \
6         --data H:glue,cola \
7         --metric H:glue,cola \
8         --figure_of_merit matthews_correlation \
9         --output_dir $1 \
10        --eqn 'classify: label ~ sentence' \
11        --num_train_epochs 3
```

GLUE COLA SUBSET

Sentence	Label
Bill sang himself to sleep.	1 (acceptable)
Bill squeezed the puppet through the hole.	1 (acceptable)
Bill sang Sue to sleep.	1 (acceptable)
The elevator rumbled itself to the ground.	0 (unacceptable)
If the telephone rang, it could ring itself silly.	1 (acceptable)
She yelled hoarse.	0 (unacceptable)
Ted cried to sleep.	0 (unacceptable)
The tiger bled to death.	1 (acceptable)

Simple Equations Cover Many Cases of Interest

GLUE: A Popular Benchmark

Subset	Dataset	Equation
COLA	H:glue,cola	<i>classify : label ~ sentence</i>
SST2	H:glue,sst2	<i>classify : label ~ sentence</i>
WNLI	H:glue,wnli	<i>classify : label ~ sentence</i>
MRPC	H:glue,mrpc	<i>classify : label ~ sentence1 + sentence2</i>
QNLI	H:glue,qnli	<i>classify : label ~ sentence1 + sentence2</i>
QQP	H:glue,qqp	<i>classify : label ~ question + sentence</i>
SSTB	H:glue,sstb	regress : <i>label ~ question1 + question2</i>
MNLI	H:glue,mnli	<i>classify : label ~ premise + hypothesis</i>

Equation Keywords \approx Pipeline Tasks

Benchmark	Subset	Dataset	Equation
GLUE	COLA	H:glue,cola	<i>classify : label ~ sentence</i>
SQuAD 1.0		H:squad	<i>classify_spans: answers ~ question + context</i>
SQuAD 2.0		H:squad_v2	<i>classify_spans: answers ~ question + context</i>
CONLL2003	POS	H:conll2003	<i>classify_tokens: pos_tags ~ tokens</i>
	NER	H:conll2003	<i>classify_tokens: ner_tags ~ tokens</i>
TIMIT		H:timit_asr	<i>ctc : text ~ audio</i>
Amazon Reviews		H:amazon_reviews_multi	<i>classify : label ~ question + sentence</i>
VAD		C:\$gft/datasets/VAD/VAD	<i>regress: Valence + Arousal + Dominance ~ Word</i>

gft Cheat Sheet

(General Fine-Tuning)

4+1 Functions

1. `gft_fit`: $f_{pre} \rightarrow f_{post}$ (fine-tuning)
 - 4 Arguments, `--output_dir`, `--metric`, `--splits`
 - (plus most args in most hubs)
2. `gft_predict`: $f(x) \rightarrow \hat{y}$ (inference)
 - Input: 4 Arguments (x from data or stdin)
 - Output: \hat{y} for each x
3. `gft_eval`: Score model on dataset
 - Input: 4 Arguments, `--split`, `--metric`, ...
 - Output: Score
4. `gft_summary`: Find good stuff
 - Input: 4 Arguments
 - (may include: `__contains__`, `__infer__`)
5. `gft_cat_dataset`: Output data to `stdout`
 - Input: 4 Arguments (`--data`, `--eqn`)

4 Arguments

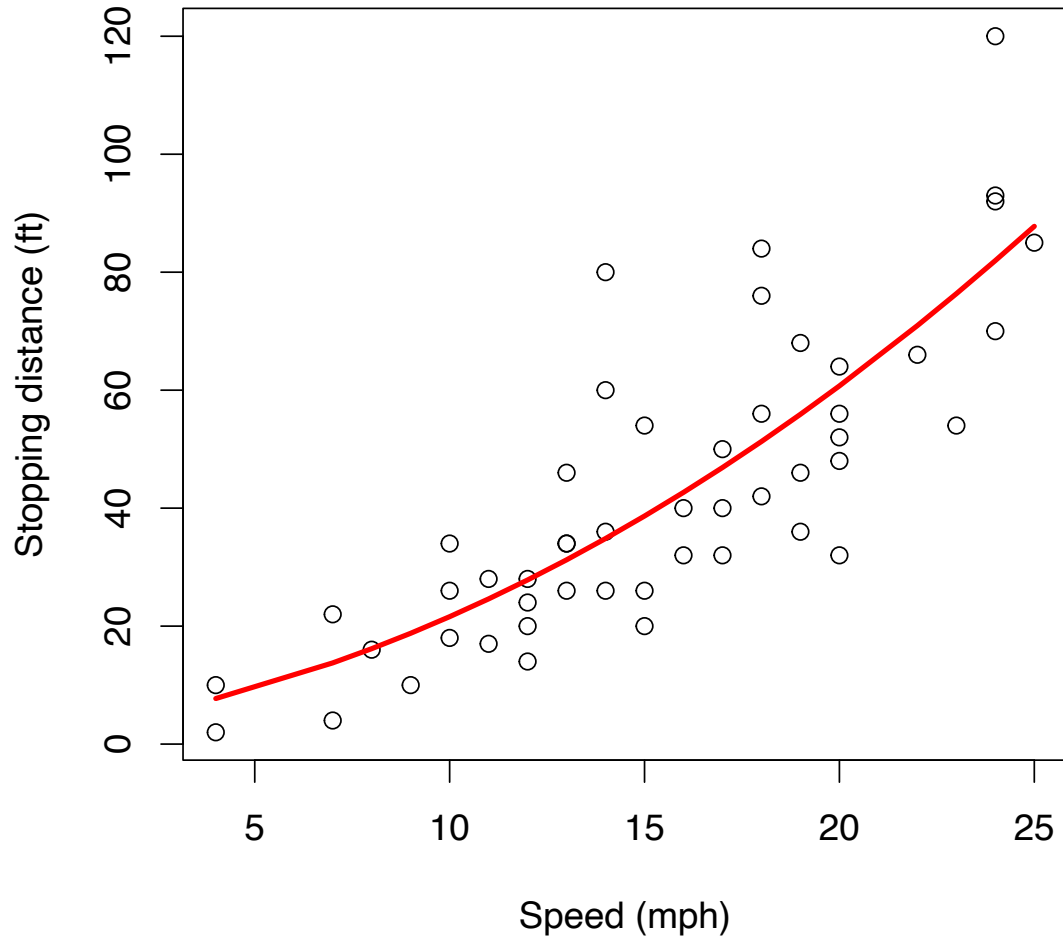
- ✓ `--data`
- ✓ `--model`
- `--eqn` *task*: $y \sim x_1 + x_2$
- `--task`
 1. `classify` (text-classification)
 2. `classify_tokens` (token-classification)
 3. `classify_spans` (QA, question-answering)
 4. `classify_audio` (audio-classification)
 5. `classify_images` (image-classification)
 6. `regress`
 7. `text-generation`
 8. `MT` (translation)
 9. `ASR` (ctc, automatic-speech-recognition)
 10. `fill-mask`

Fit a model (g) to data ($cars$)

Regression in R:

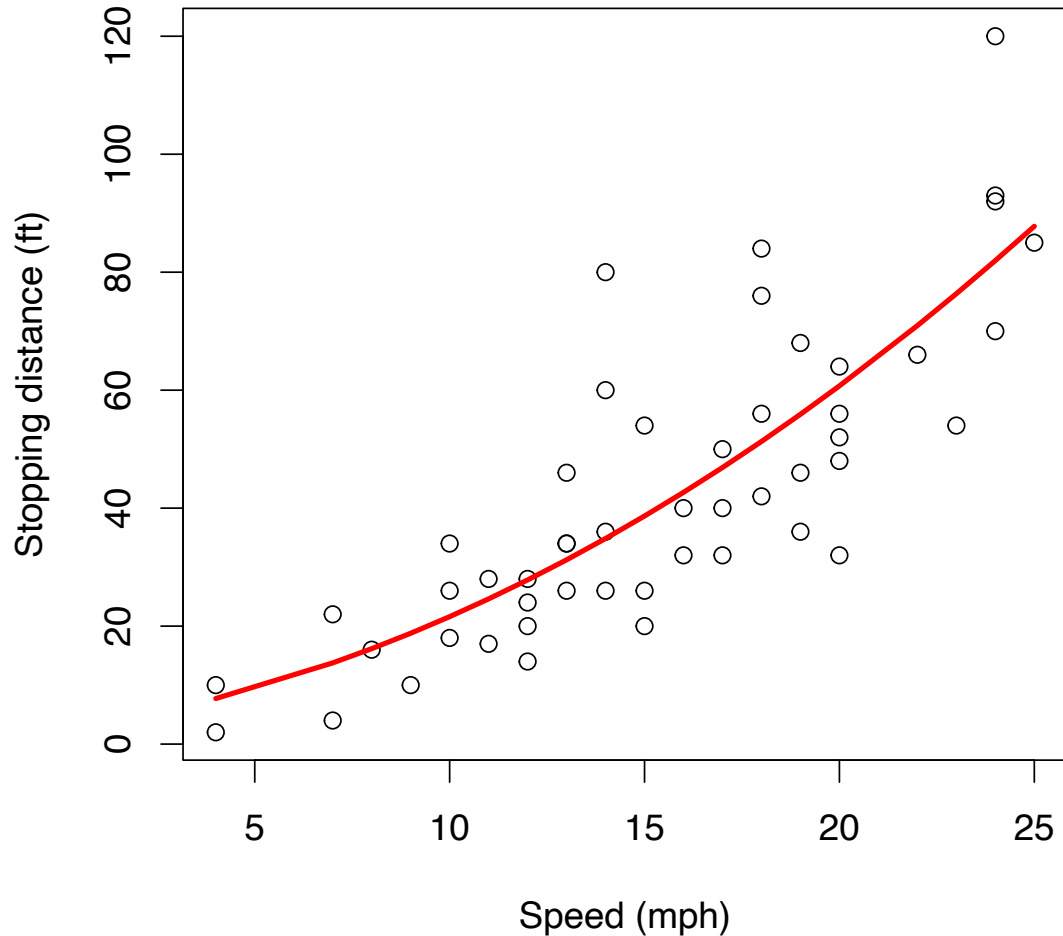
Summarize (almost) anything

```
> plot(cars, xlab="Speed (mph)", ylab="Stopping distance (ft)")  
> g = glm(dist ~ poly(speed,2), data=cars)  
> o = order(cars$speed)  
> lines(cars$speed[o], predict(g,cars)[o], col="red", lwd=3)
```



Regression in R:

Summarize (almost) anything



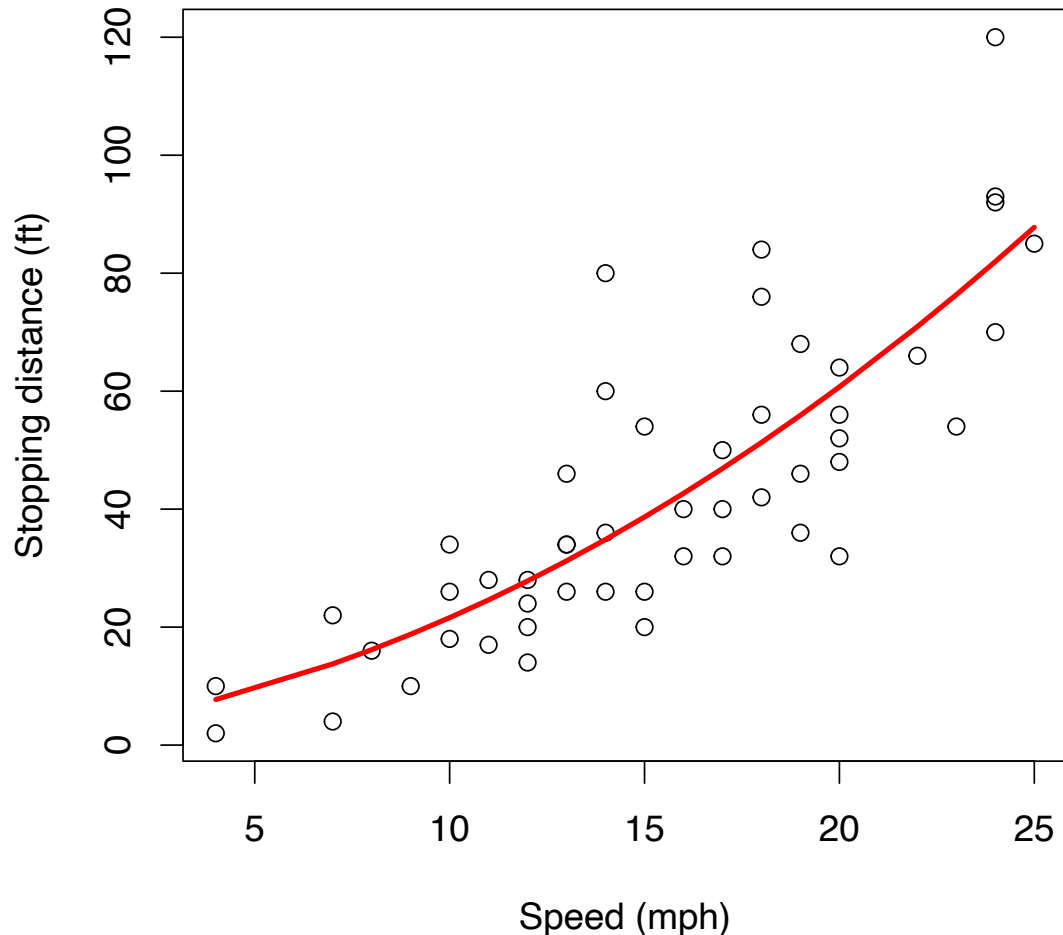
```
> summary(cars)
```

speed	dist
Min. : 4.0	Min. : 2.00
1st Qu.: 12.0	1st Qu.: 26.00
Median : 15.0	Median : 36.00
Mean : 15.4	Mean : 42.98
3rd Qu.: 19.0	3rd Qu.: 56.00
Max. : 25.0	Max. : 120.00

Fit a model (g) to data ($cars$)

Regression in R:

Summarize (almost) anything



```
> plot(cars, xlab="Speed (mph)", ylab="Stopping distance (ft)")
> g = glm(dist ~ poly(speed,2), data=cars)
> o = order(cars$speed)
> lines(cars$speed[o], predict(g,cars)[o], col="red", lwd=3)
> summary(g)
```

Predict

Summarize a model (g)

Call:

```
glm(formula = dist ~ poly(speed, 2), data = cars)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-28.720	-9.184	-3.188	4.628	45.152

Opportunity to Improve g

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	42.980	2.146	20.026	< 2e-16	***
poly(speed, 2)1	145.552	15.176	9.591	1.21e-12	***
poly(speed, 2)2	22.996	15.176	1.515	0.136	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 230.3131)

Null deviance: 32539 on 49 degrees of freedom
Residual deviance: 10825 on 47 degrees of freedom
AIC: 418.77

Number of Fisher Scoring iterations: 2

gft_summary

- Summarize almost anything
 - Models
 - Datasets

Tasks

• classify, text-classification

$y \in \{0,1,2, \dots\}$

• regress

$y \in \mathbb{R}$ or $y \in \mathbb{R}^N$

• QA, Question Answering, classify spans

y for each start/end of span

• token classification

y for each token

- NER (Named Entity Recognition)
- POS (Part of Speech Tagging)

• translation, MT

• ASR, Automatic Speech Recognition, etc

y for each phoneme

gft Cheat Sheet

(General Fine-Tuning)

4+1 Functions

1. `gft_fit`: $f_{pre} \rightarrow f_{post}$ (fine-tuning)
 - 4 Arguments, `--output_dir`, `--metric`, `--splits`
 - (plus most args in most hubs)
2. `gft_predict`: $f(x) \rightarrow \hat{y}$ (inference)
 - Input: 4 Arguments (x from data or stdin)
 - Output: \hat{y} for each x
3. `gft_eval`: Score model on dataset
 - Input: 4 Arguments, `--split`, `--metric`, ...
 - Output: Score
4. `gft_summary`: Find good stuff
 - Input: 4 Arguments
 - (may include: `__contains__`, `__infer__`)
5. `gft_cat_dataset`: Output data to `stdout`
 - Input: 4 Arguments (`--data`, `--eqn`)

4 Arguments

- ✓ `--data`
- ✓ `--model`
- ✓ `--eqn` *task*: $y \sim x_1 + x_2$
- ✓ `--task`
 1. `classify` (text-classification)
 2. `classify_tokens` (token-classification)
 3. `classify_spans` (QA, question-answering)
 4. `classify_audio` (audio-classification)
 5. `classify_images` (image-classification)
 6. `regress`
 7. `text-generation`
 8. `MT` (translation)
 9. `ASR` (ctc, automatic-speech-recognition)
 10. `fill-mask`

Machine Learning Before Deep Nets

- Collocation
 - PMI (Church & Hanks, 1990)
- Word2vec (Factored PMI)
- Anything to vec
 - Homework: Better Together
 - Doc2vec (in many ways)
 - Cosine similarity
 - Similar text vs. similar context
- Regression: linear, logistic
- Sklearn and Scipy

6 CHAPTER 6 • VECTOR SEMANTICS AND EMBEDDINGS



Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from [Li et al. \(2015\)](#) with colors added for explanation.

JM3 Chapter 6 (Spoiler Alert: Reading for next time)

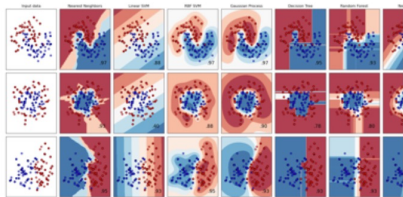
Sklearn



Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...



Examples

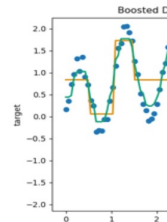
Dimensionality reduction

Reducing the number of random variables to

Regression

Predicting a continuous value with an object.

Applications: Drug re
Algorithms: Gradient random forest, ridge, z



Model selectio

Comparing, validating



Prev Up Next

scikit-learn 1.3.0
Other versions

Please **cite us** if you use the software.

User Guide

1. Supervised learning
2. Unsupervised learning
3. Model selection and evaluation
4. Inspection
5. Visualizations
6. Dataset transformations
7. Dataset loading utilities
8. Computing with scikit-learn
9. Model persistence
10. Common pitfalls and recommended practices
11. Dispatching

User Guide

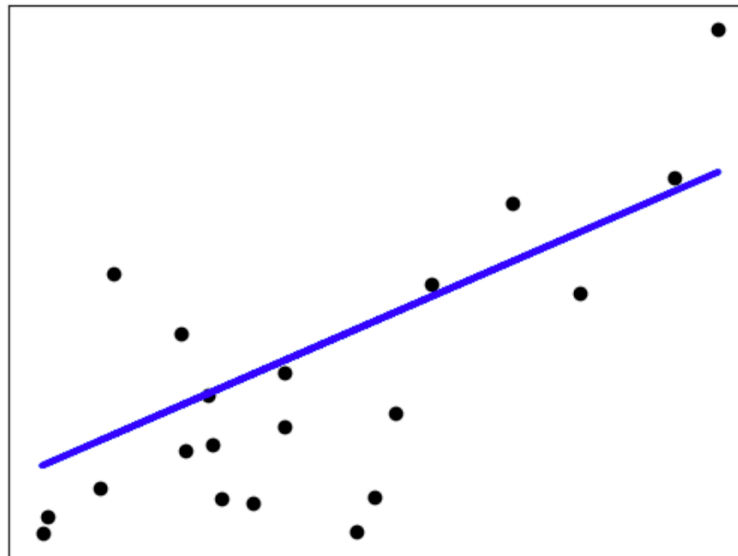
1. Supervised learning

- ▶ 1.1. Linear Models
- ▶ 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- ▶ 1.4. Support Vector Machines
- ▶ 1.5. Stochastic Gradient Descent
- ▶ 1.6. Nearest Neighbors
- ▶ 1.7. Gaussian Processes
- ▶ 1.8. Cross decomposition
- ▶ 1.9. Naive Bayes
- ▶ 1.10. Decision Trees
- ▶ 1.11. Ensemble methods
- ▶ 1.12. Multiclass and multioutput algorithms
- ▶ 1.13. Feature selection

Linear Regression Example

The example below uses only the first feature of the `diabetes` dataset, in order to illustrate the data points within the two-dimensional plot. The straight line can be seen in the plot, showing how linear regression attempts to draw a straight line that will best minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation.

The coefficients, residual sum of squares and the coefficient of determination are also calculated.



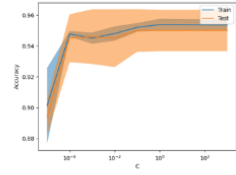
Prev Up Next

scikit-learn 1.3.0
Other versions

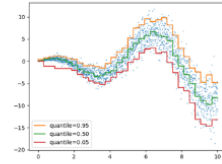
Please cite us if you use the software.

skLearn.linear_model.LogisticRegression
LogisticRegression
Examples using
skLearn.linear_model.LogisticRegression

Examples using `sklearn.linear_model.LogisticRegression`



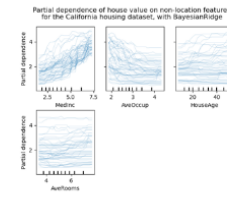
Release Highlights for scikit-learn 1.3



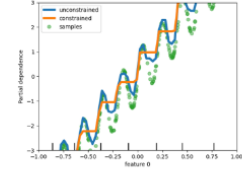
Release Highlights for scikit-learn 1.1



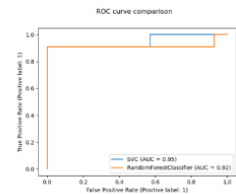
Release Highlights for scikit-learn 1.0



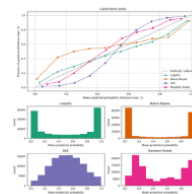
Release Highlights for scikit-learn 0.24



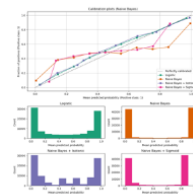
Release Highlights for scikit-learn 0.23



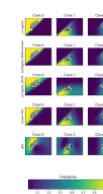
Release Highlights for scikit-learn 0.22



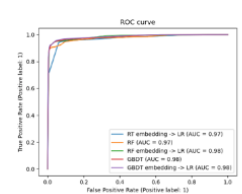
Comparison of Calibration of Classifiers



Probability Calibration curves



Plot classification probability



Feature transformations with ensembles of trees

Linear Regression and Logistic Regression

- Linear

- `g1 = glm(dist ~ poly(speed,2), data=cars)`

- Logistic

- `g2 = glm(dist > 50 ~ poly(speed,2), data=cars, family=binomial)`

- Models

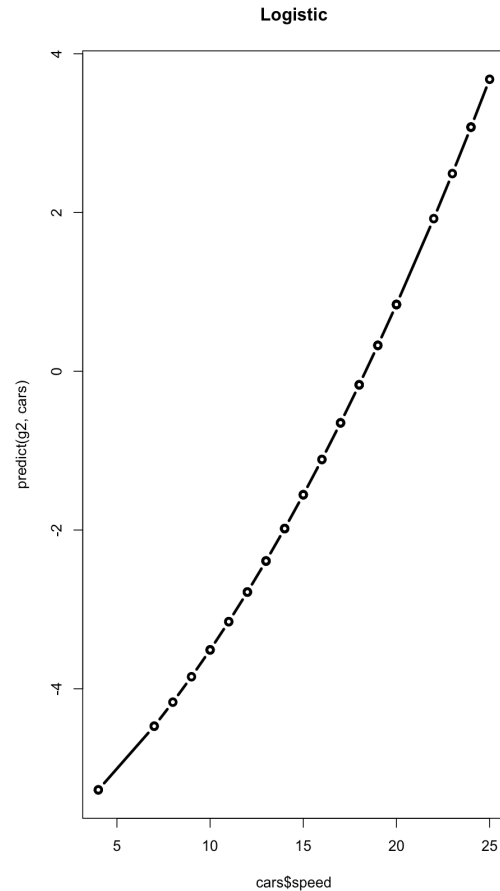
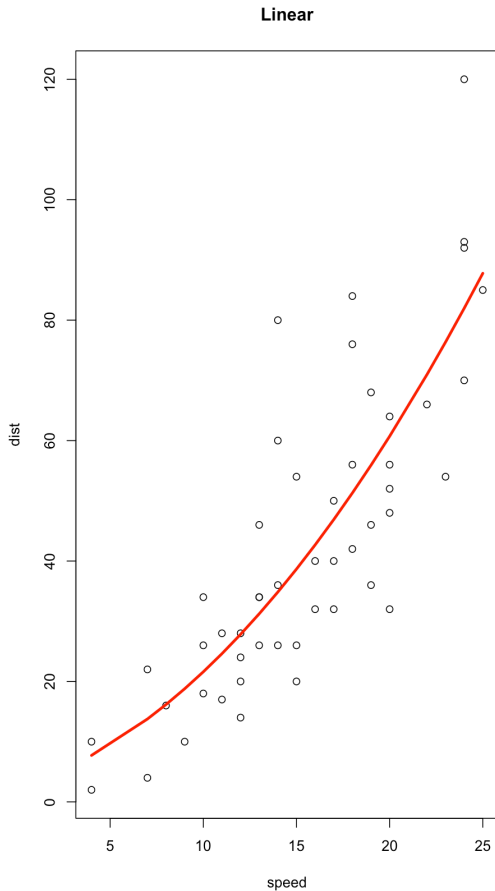
- > `g1$coef`

- (Intercept) poly(speed, 2)1 poly(speed, 2)2
 - 42.98000 145.55226 22.99576

- > `g2$coef`

- (Intercept) poly(speed, 2)1 poly(speed, 2)2
 - 1.137922 16.172581 2.031136

Linear and Logistic Regression



```
> g1 = glm(dist ~ poly(speed,2),  
data=cars)
```

```
> g2 = glm(dist > 50 ~  
poly(speed,2), data=cars,  
family=binomial)
```

```
> plot(cars, main="Linear")
```

```
> lines(cars$speed, predict(g1,  
cars), col="red", lwd=3)
```

```
> plot(cars$speed, predict(g2,  
cars), type='b', lwd=3,  
main="Logistic")
```

Inverses

`sigmoid = function(z) 1 / (1 + exp(-z))`

`logit = function(p) log(p / (1 - p))`

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\text{logit}(p) = \log \frac{p}{1 - p}$$

Note: sigmoid & logit are inverses of one another:

$$\text{logit}(\sigma(z)) = z$$

$$\sigma(\text{logit}(p)) = p$$

Linear and Logistic Regression

```
> g1 = glm(dist ~  
poly(speed,2), data=cars)  
> g2 = glm(dist > 50 ~  
poly(speed,2), data=cars,  
family=binomial)
```

```
> plot(cars$speed,  
predict(g2, cars),  
type='b', lwd=3,  
main="Without Sigmoid",  
ylab="Logit")
```

```
> plot(cars$speed,  
sigmoid(predict(g2,  
cars)), type='b', lwd=3,  
main="With Sigmoid",  
ylab="Probability")
```

Recap

- What's where:
 - <https://kwchurch.github.io/>
- Formalities:
 - Northeastern Rules, Culture, etc.
 - Instructors, TA, Office Hours
- Goals, Expectations, etc.
 - Encouraging learning:
 - Teamwork, Online Resources, Bots
 - As long as these tools → learning
 - Grading:
 - Assignments:
 - Effort/Learning >> Solving Problems
 - Mid-Term Exam:
 - Open book, network
 - But no teams
 - Final Project: Proposal, Oral, Written
- Better Together Homework
- Little Languages: Unix, AWK, R
- Deep Nets
 - Inference (predict)
 - Fine-Tuning (fit)
 - Pre-training
 - aka base/fundamental models
 - (don't do it)
- Deep nets are like regression
 - Linear regression:
 - Cars example: fit/predict in R
 - Datasets:
 - cars dataset in R is like
 - emotion dataset on huggingface
 - Classification vs. Regression
 - Logistic Regression: example of binary classification
 - Similarities of linear and logistic regression

Recap

- What's where:
 - <https://kwchurch.github.io/>
- Formalities:
 - Northeastern Rules, Culture, etc.
 - Instructors, TA, Office Hours
- Goals, Expectations, etc.
 - Encouraging learning:
 - Teamwork, Online Resources, Bots
 - As long as these tools → learning
 - Grading:
 - Assignments:
 - Effort/Learning >> Solving Problems
 - Mid-Term Exam:
 - Open book, network
 - But no teams
 - Final Project: Proposal, Oral, Written

Recap

- What's where:
 - <https://kwchurch.github.io/>
- Formalities:
 - Northeastern Rules, Culture, etc.
 - Instructors, TA, Office Hours
- Goals, Expectations, etc.
 - Encouraging learning:
 - Teamwork, Online Resources, Bots
 - As long as these tools → learning
 - Grading:
 - Assignments:
 - Effort/Learning >> Solving Problems
 - Mid-Term Exam:
 - Open book, network
 - But no teams
 - Final Project: Proposal, Oral, Written

SciPy