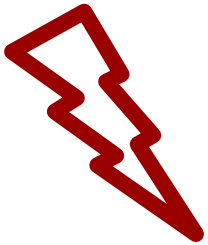
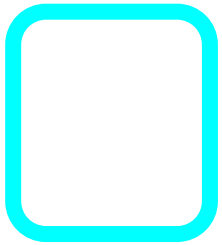
k-Nearest Neighbors

Samuel Taylor


http://www.logodust.com/

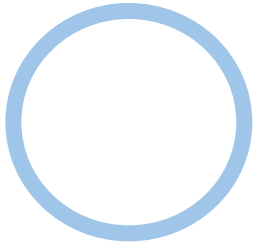


Chervny



Modry

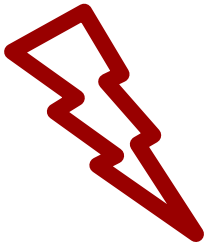
Chervny



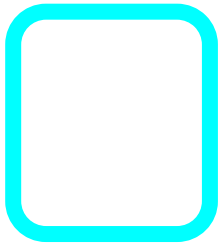
Modry

Chervny


Yo
It's that simple.
Download it from
App Store
Download it from
Google Play

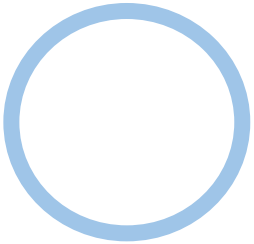


Chervny



Modry

Chervny



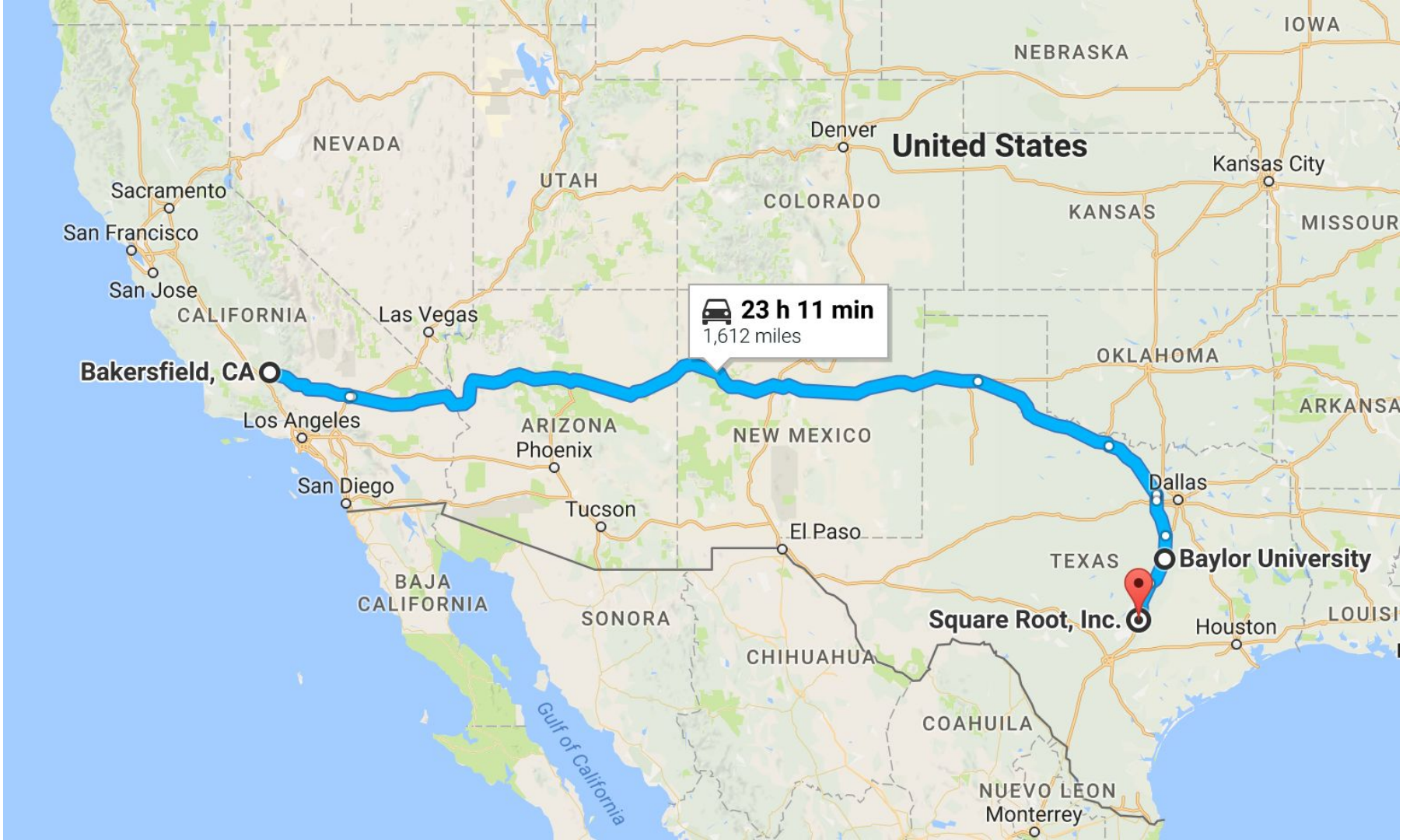
Modry

Chervny

?

Outline

- **Introduction**
- Algorithm
- How good is it?
- Application


Bakersfield, CA
Baylor University
Square Root, Inc.
23 h 11 min
1,612 miles
United States
Denver
Kansas City
Sacramento
San Francisco
San Jose
Las Vegas
Los Angeles
Phoenix
San Diego
Tucson
El Paso
Dallas
Houston
Monterrey
NEVADA
UTAH
COLORADO
KANSAS
NEBRASKA
IOWA
CALIFORNIA
ARIZONA
NEW MEXICO
OKLAHOMA
TEXAS
BAJA CALIFORNIA
SONORA
CHIHUAHUA
COAHUILA
NUEVO LEON
Gulf of California

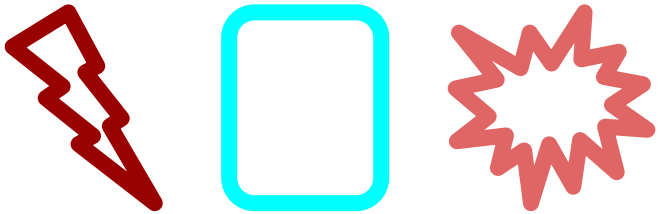
Are you familiar with...

- Approximation–generalization tradeoff?
- VC dimension?
- Regularization?

Outline

- Introduction
- **Algorithm**
- How good is it?
- Application

How did we do that?

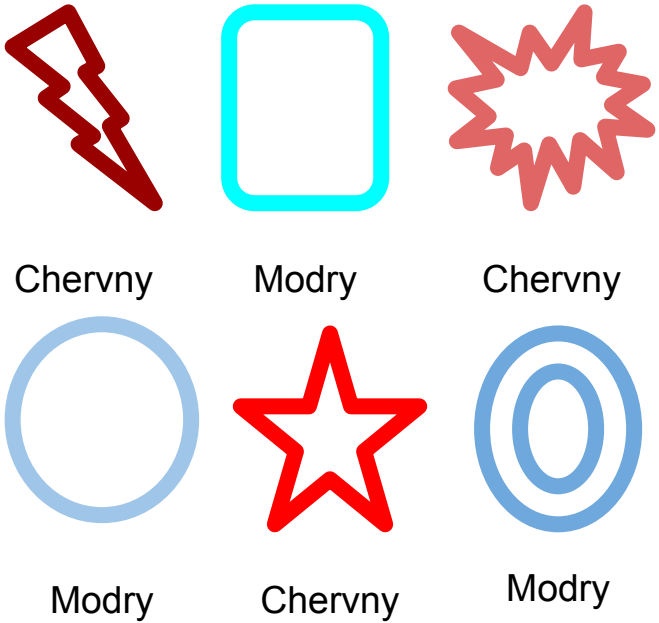


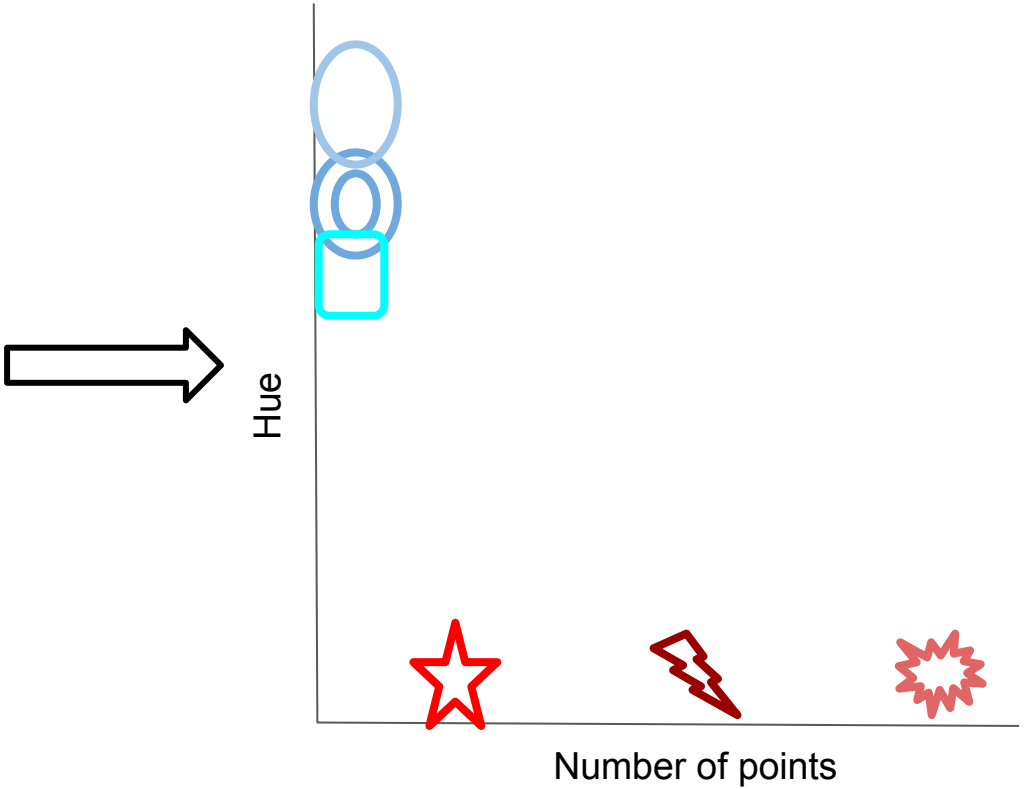
Chervny Modry Chervny

Modry Chervny Modry

- Observe training data
- Find most similar datum
  - Color
  - Points
- Pick the class of the similar datum

What does similar mean?





The Nearest Neighbor Algorithm

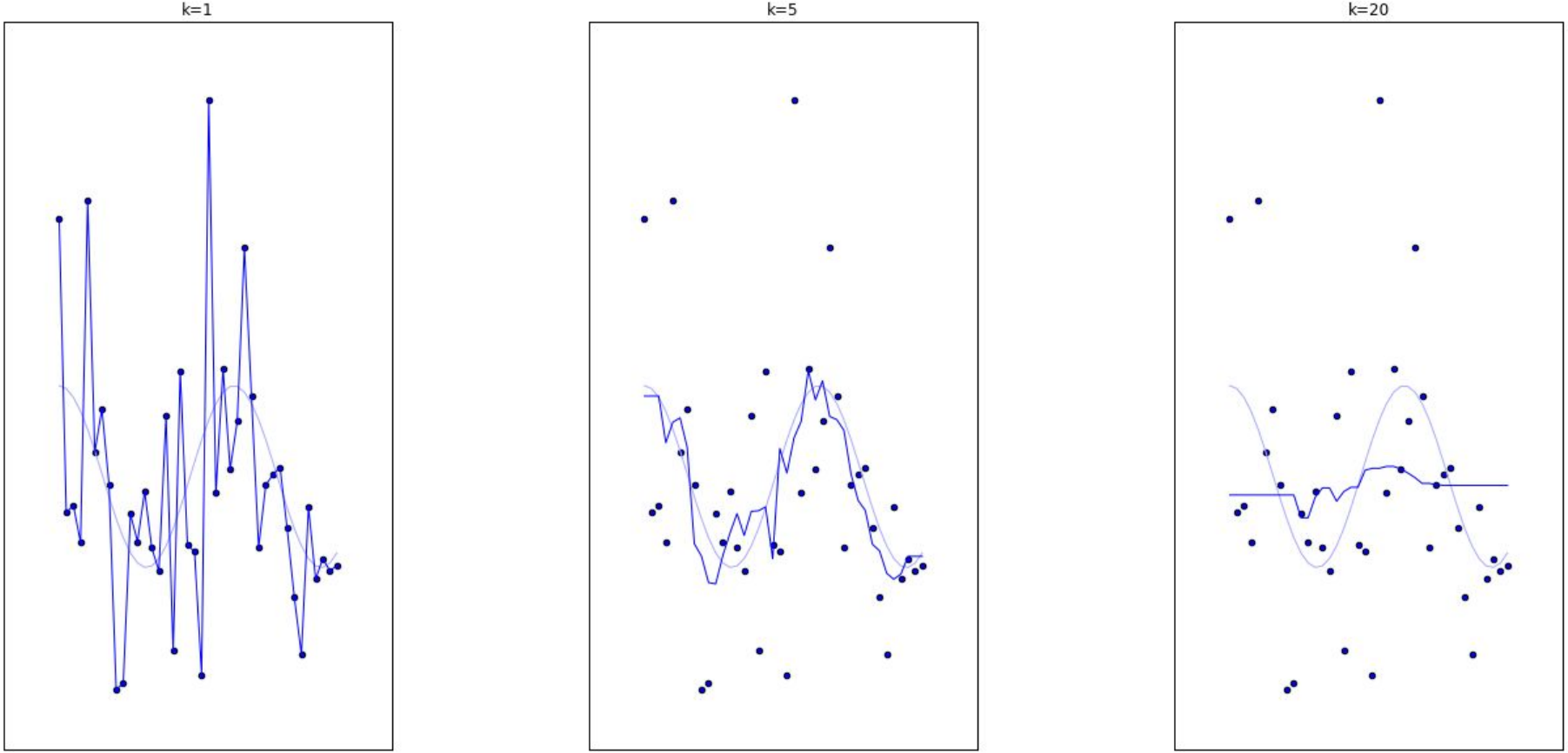
- Given a training set $\boldsymbol{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ and a data point u,
- Find the point $d = (x_j, y_j)$ in $\boldsymbol{D}$ for which x_j is closest to u
- Return y_j

The *k*-Nearest Neighbor Algorithm

- Given a training set $\boldsymbol{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ and a data point u,
- Find the k nearest points to u $(x_{n1}, y_{n1}), \ldots, (x_{nk}, y_{nk})$ in $\boldsymbol{D}$
- Return sign(sum(y_{ni} for i in range(k)) / k)

Observations

- No training required
- Low values of *k* tend toward overfit, high values tend to underfit


k=1
k=5
k=20

Observations

- Very low in-sample error
- Infinite VC-dimension

Outline

- Introduction
- Algorithm
- **How good is it?**
- Application

Pretty good

- ≤ 2 × minimum possible out of sample error
  - Probably, as N -> ∞
  - Under reasonable assumptions

Proving $E_{\text{out}} \leq 2E^*_{\text{out}}$

$$\pi(\mathbf{x}) = \mathbb{P}[y = +1|\mathbf{x}].$$ ← the target in logistic regression

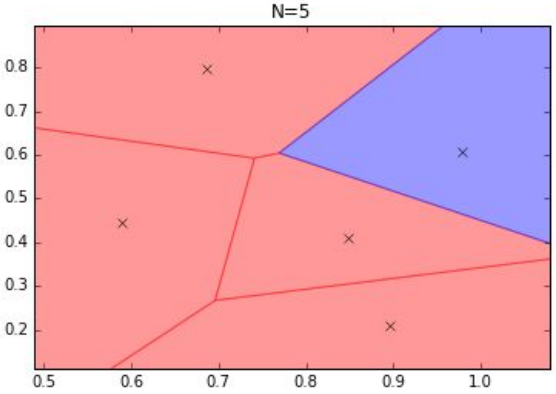
Assume $\pi(\mathbf{x})$ is continuous and $\mathbf{x}_{[1]} \overset{N\to\infty}{\longrightarrow} \mathbf{x}$. Then $\pi(\mathbf{x}_{[1]}) \overset{N\to\infty}{\longrightarrow} \pi(\mathbf{x})$.

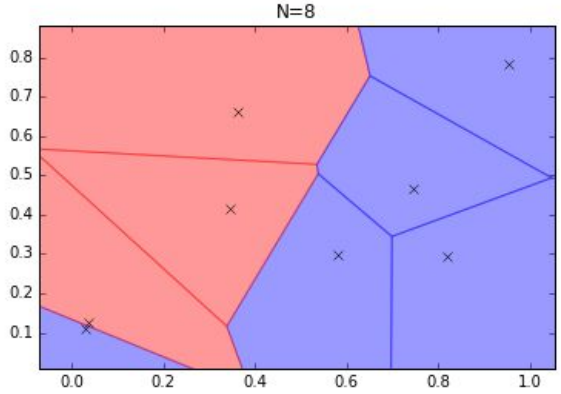
$$\begin{aligned}
\mathbb{P}[g_N(\mathbf{x}) \neq y] &= \mathbb{P}[y = +1, y_{[1]} = -1] \; + \; \mathbb{P}[y = -1, y_{[1]} = +1], \\
&= \pi(\mathbf{x}) \cdot (1 - \pi(\mathbf{x}_{[1]})) + (1 - \pi(\mathbf{x})) \cdot \pi(\mathbf{x}_{[1]}), \\
&\to \pi(\mathbf{x}) \cdot (1 - \pi(\mathbf{x})) + (1 - \pi(\mathbf{x})) \cdot \pi(\mathbf{x}), \\
&= 2\pi(\mathbf{x}) \cdot (1 - \pi(\mathbf{x})), \\
&\leq 2 \min\{\pi(\mathbf{x}), 1 - \pi(\mathbf{x})\}.
\end{aligned}$$

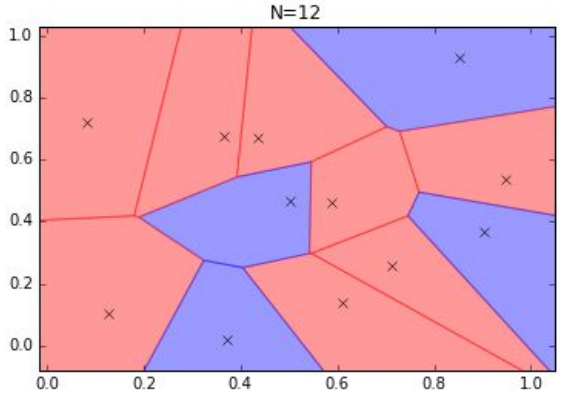
The best you can do is

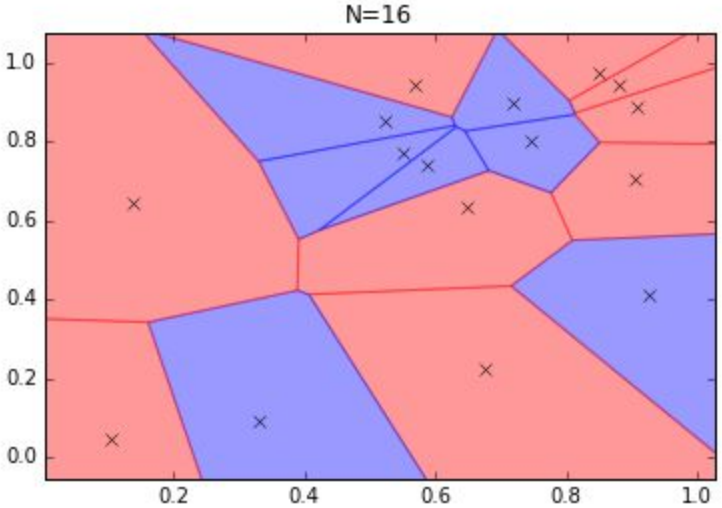
$$E^*_{\text{out}}(\mathbf{x}) = \min\{\pi(\mathbf{x}), 1 - \pi(\mathbf{x})\}.$$

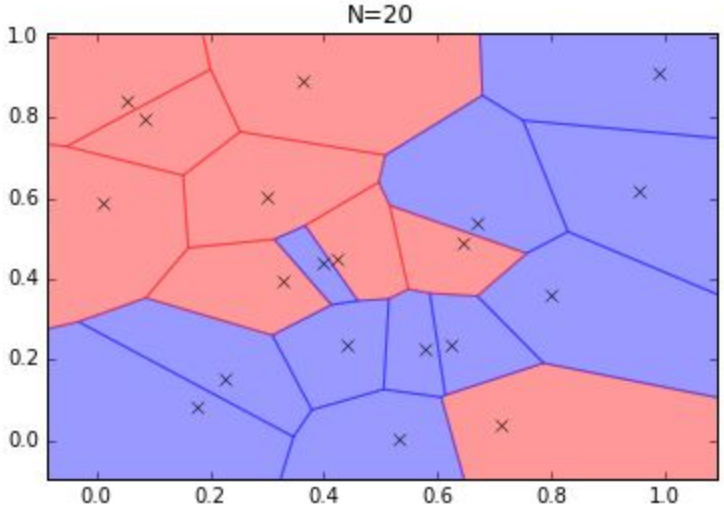
NN is self-regularizing


N=5


N=8

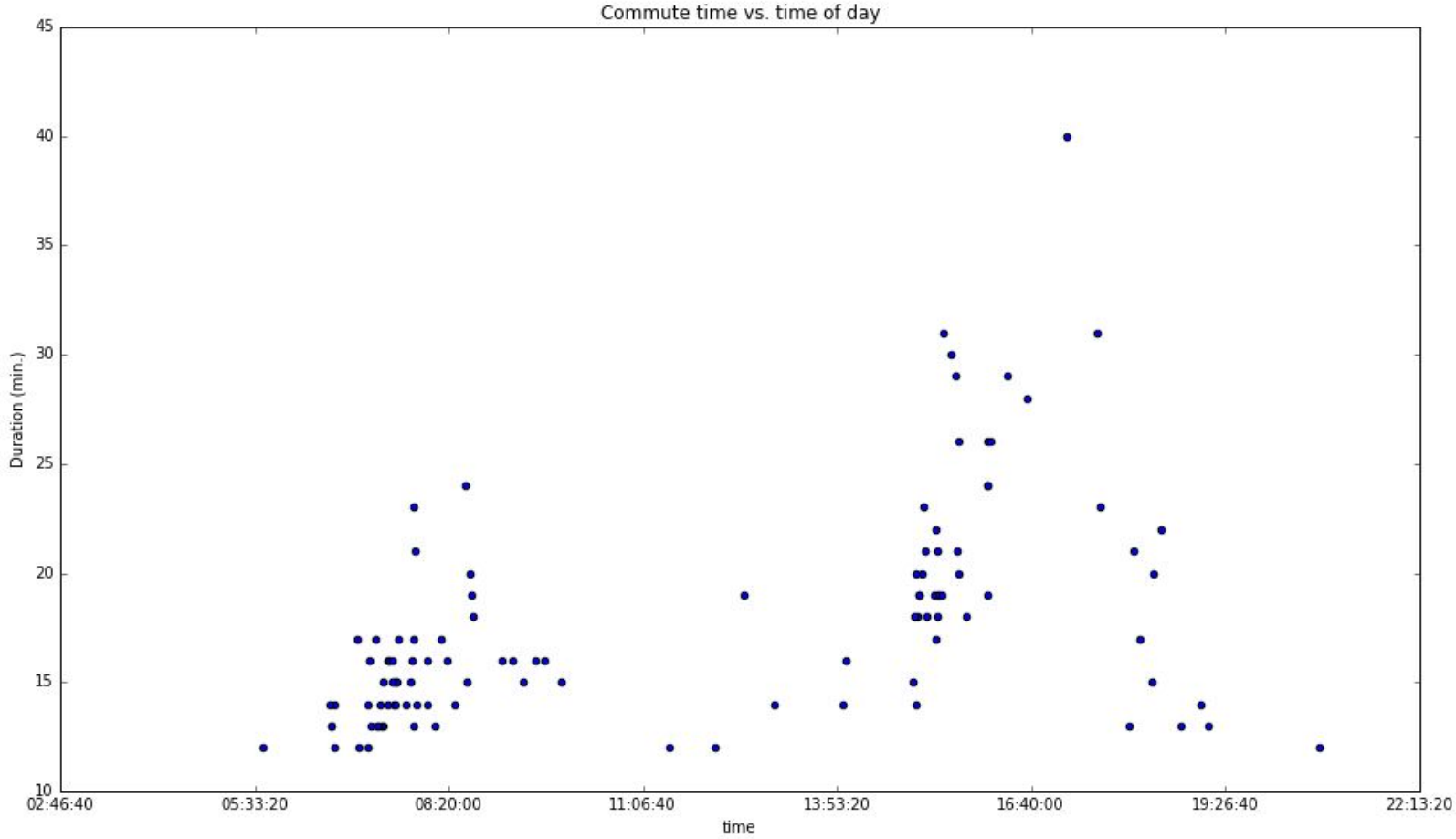

N=12


N=16

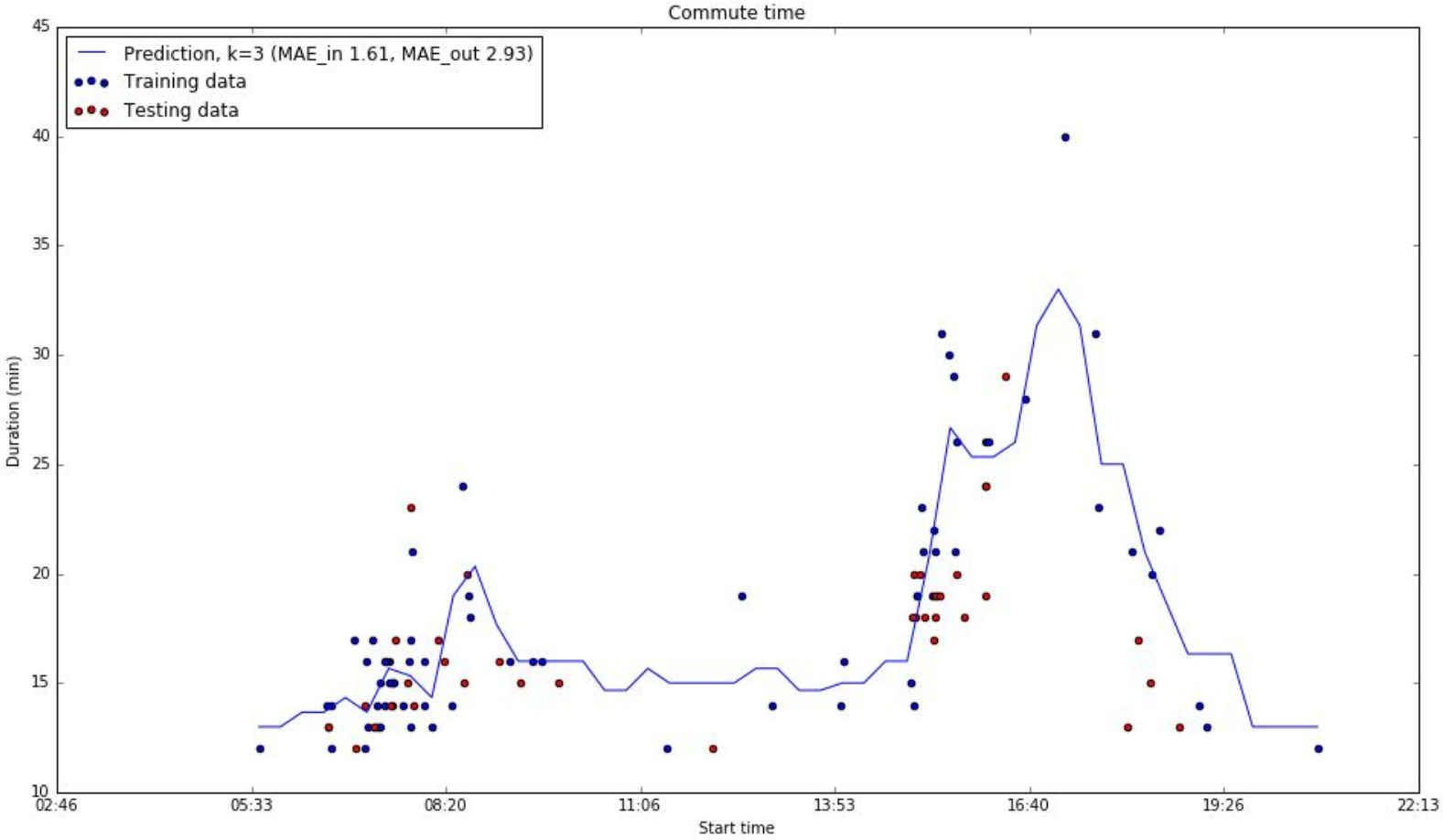

N=20

Outline

- Introduction
- Algorithm
- How good is it?
- **Application**


Commute time vs. time of day
Duration (min.)
45
40
35
30
25
20
15
10
02:46:40
05:33:20
08:20:00
11:06:40
13:53:20
16:40:00
19:26:40
22:13:20
time

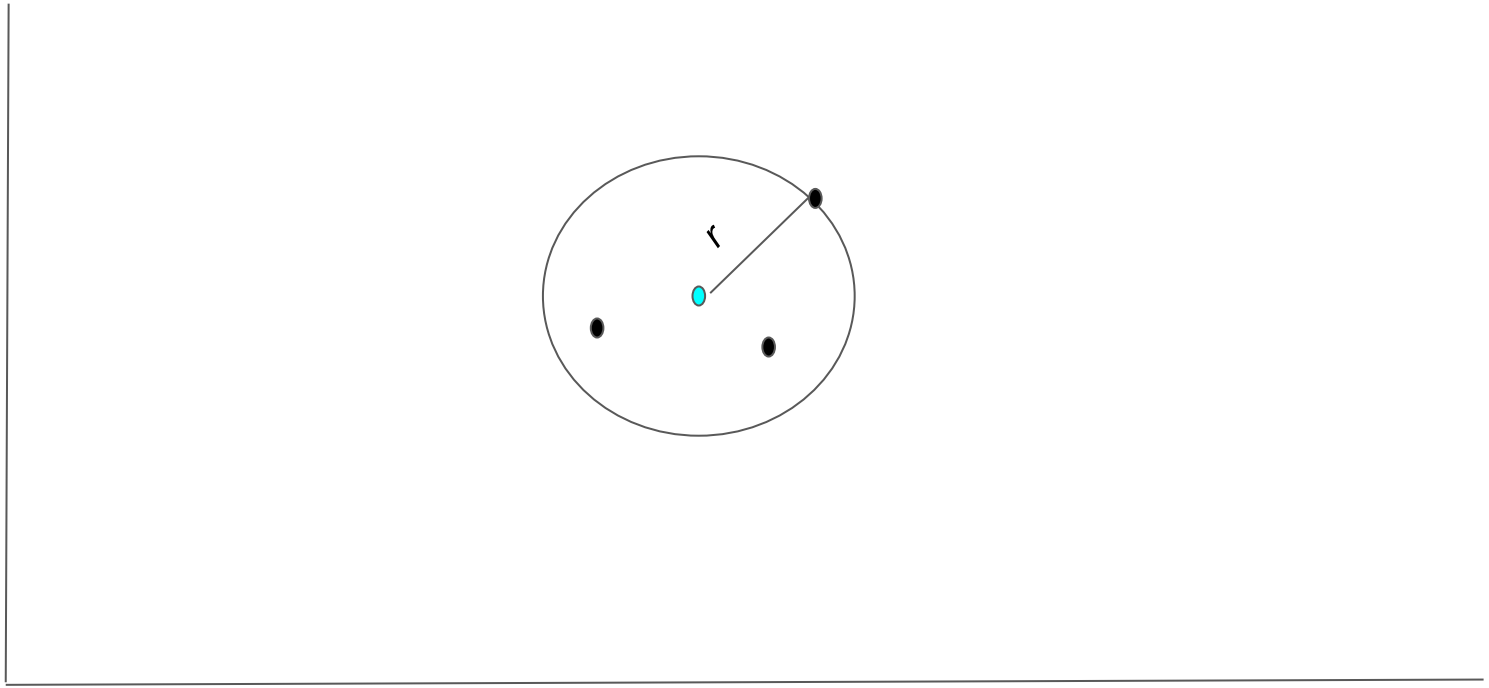
```
url = 'http://dash.samueltaylor.org/commute_time.csv'
commutes = pd.read_csv(url, index_col=0, parse_dates=True)
X = 60 * (commutes.index.hour * 60 + commutes.index.minute)
y = commutes['duration']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33)

regr = neighbors.KNeighborsRegressor(3)
regr.fit(X_train, y_train)

predictions = regr.predict(X_test)
```

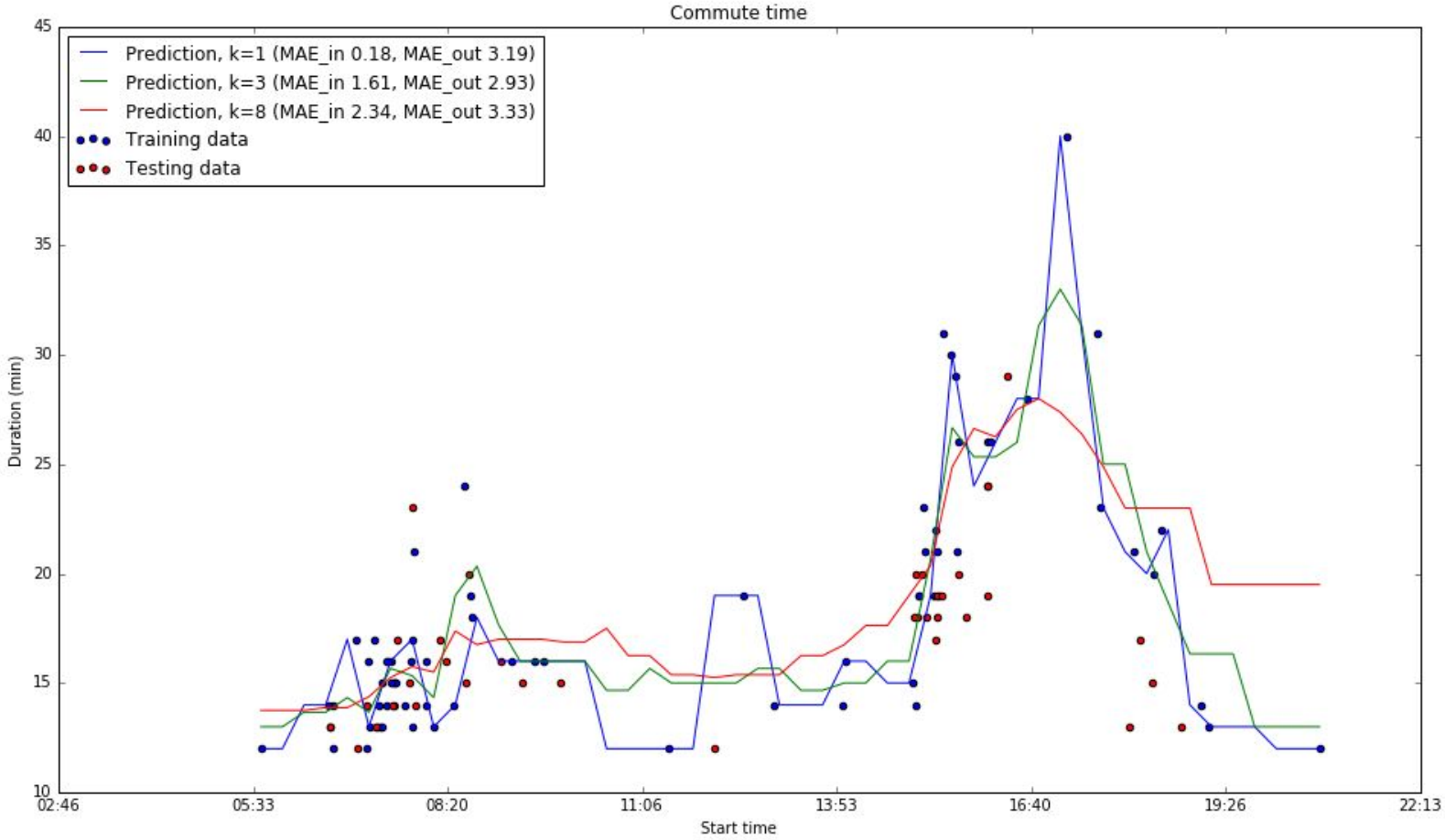

Commute time
Prediction, k=3 (MAE_in 1.61, MAE_out 2.93)
Training data
Testing data
Duration (min)
45
40
35
30
25
20
15
10
02:46
05:33
08:20
11:06
13:53
16:40
19:26
22:13
Start time

Choosing *k*

- Controls the tradeoff between approximation and generalization
- 3
- floor(sqrt(N))
  - As $N \to \infty$ and $k(N)/N \to 0$, out of sample error approaches optimal

Choosing *k*




Commute time
Prediction, k=1 (MAE_in 0.18, MAE_out 3.19)
Prediction, k=3 (MAE_in 1.61, MAE_out 2.93)
Prediction, k=8 (MAE_in 2.34, MAE_out 3.33)
Training data
Testing data
Duration (min)
45
40
35
30
25
20
15
10
02:46
05:33
08:20
11:06
13:53
16:40
19:26
22:13
Start time

Choosing *k*

- Controls the tradeoff between approximation and generalization
- 3
- floor(sqrt(N))
  - As $N \to \infty$ and $k(N)/N \to 0$, out of sample error approaches optimal
- Cross validation

Unaddressed questions

- How do I choose a similarity function?
- How do I choose features?
- Why isn’t the scikit implementation terribly slow (esp. on large datasets)?

Further reading

- *Learning From Data*; Abu-Mostafa, Magdon-Ismail, and Lin
- Similarity and Nearest Neighbor; Malik Magdon-Ismail

Tally raises $15 million for ...

https://techcrunch.com/2016/05/19/tally-raises-15-million-for-app-to-make-credit-cards

Search

Got a tip? Let us know.

News Video Events Crunchbase

Search

Apps | cowboy ventures | shasta ventures

ShapeCo raises $15 million for app to tell people what shapes are

Posted May 19, 2016 by Lora Kolodny (@lorakolodny)

Austin, Texas-based ShapeCo raised $15 million in Series A venture funding to launch an app that promises to help people be better at understanding shapes. Shasta Ventures led the Series A and was joined by the company's earlier backers — Cowboy Ventures and AITV. Silicon Valley Bank also invested.

According to ShapeCo CEO and co-founder Chester Snapdragon McFisticuff, the company has been operating in stealth for longer than a year and has been testing its service with beta

Crunchbase

Cowboy Ventures −

FOUNDED
2012

OVERVIEW
Cowboy Ventures is a seed-stage focused fund. We seek to back exceptional founders who are building products that "re-imagine" work and personal life in large and growing markets – we call it "Life 2.0". For more information, see www.cowboy.vc

LOCATION
Palo Alto, CA

CATEGORIES
Venture Capital

FOUNDERS
Aileen Lee

WEBSITE
http://www.cowboy.vc

Full profile for Cowboy Ventures

Aileen Lee +

Samuel Taylor

kdd@samueltaylor.org