

Put content on this frame that directs them to binder or
colab.research.google.com

Before we get going here, I'd like to direct you to colab.research.google.com to get things set up for the live coding demo that will take place in the second half of our time today. Follow the instructions on my github at...

Machine Learning From Scratch

Collin Prather

April 25th, 2018

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Part 1) Applied Machine Learning case study: Drunk Driving Car Crashes

Part 2) [We totally change gears] SVM from scratch that exemplifies/applies the general math process that each ML project follows (by that I mean: Representation, Evaluation, Optimization)
The main goal here is to expose you to a wide spectrum of technical/theoretical machine learning considerations in as applied of a setting as possible.

Good morning, and thank you all for making it out this early! These conference days are long, and to be here bright and early on day 3 is no joke, so I'm glad you're here.

We are going to be tackling machine learning from a bottom-up approach today. Can I get a quick show of hands, how many of us here have experience applying machine learning to solving problems?

Before we really dive in, I want to tell you a bit about my talk today, and to do so, we're going to start with two things that help tremendously in successfully applying machine learning – (1) Understanding the math at

Special Thanks

Dr. D

Dr. Michael Bloem

└ Special Thanks

Dr. D
Dr. Michael Bloem

I'd like to offer a very special thank you to Dr. Devereaux and Dr. Michael Bloem for all that they've done to help me in putting this together – would not be possible without them!

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Machine Learning From Scratch

Machine Learning Overview

Machine Learning Overview

Steps in the Machine Learning Process

- Step 0: Identify The Problem
- Step 1: Get the Data
- Step 2: Data Exploration
- Step 3: Data Preparation
- Step 4: Model Selection
- Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

- Representation
- Evaluation
- Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Pull a graph of google search trends indicating how terms like "Data Science" and "Machine Learning" have blown up.

Try to form talk around hitting on the theoretical mathematical side of ML as well as the difficulties/complexities faced in Applied ML

data + algorithms = predicting the future (it's really a lot more than this – understanding context and how to frame the question (usually) from a business perspective is huge)

classification v. regression

supervised/unsupervised/reinforcement learning

when talking on reinforcement learning, mention and recommend

AlphaGo documentary (it's on netflix!)

considerations/complexities in building ML models

What is Machine Learning?



Machine Learning

Arthur Samuel:

Machine learning is “Field of study that gives computers the ability to learn without being explicitly programmed” .

Machine Learning From Scratch

└ Machine Learning Overview

└ What is Machine Learning?

What is Machine Learning?



Machine Learning

Arthur Samuel:

Machine learning is "Field of study that gives computers the ability to learn without being explicitly programmed".

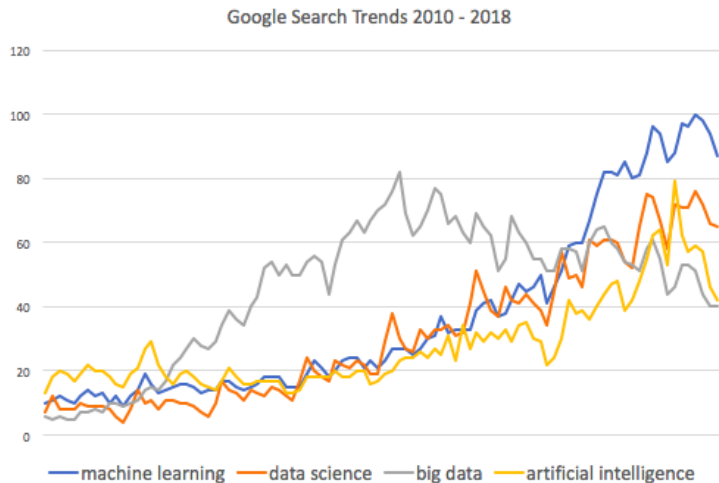
All this talk begs the question, what is machine learning? Even according to the experts, the exact definition of the field of machine learning is a bit fuzzy, but As early as 1959, Arthur Samuel was famously quoted as defining machine learning as...

ML techniques can be applied to a wide range of problems in diverse industries. In fact, ML has become ubiquitous in our everyday lives

- * Siri/ Amazon Alexa
- * Recommendation systems (amazon, netflix)
- * Fraud Detection
- * Disease diagnosis
- * Supply Chain Optimization

The list goes on and on. With sufficient data, ML algorithms combines statistics and calculus and can often make very accurate predictions about the future.

According to Google...



2018-09-05

Machine Learning From Scratch

└ Machine Learning Overview

└ According to Google...

According to Google...



This may not be a surprise to you, but the world's google search history reflects a steady increase in interest in machine learning and other related terms like Data Science, Big Data, and Artificial Intelligence.

This graph was pulled pretty simply from trends.google.com, which is actually fascinating, they make it very simple to look up search history trends.

What has caused this spike?

1. Data Availability

- ▶ digital data
- ▶ IoT (sensor data)

2. Computational Scale

- ▶ Moore's Law

Machine Learning From Scratch

└ Machine Learning Overview

└ What has caused this spike?

What has caused this spike?

1. Data Availability
 - digital data
 - IoT (sensor data)
2. Computational Scale
 - Moore's Law

What has caused this spike? The math that powers machine learning algorithms has been around for quite a few years... so what's changed? It really boils down to two things.

1. Data Availability (here we should just a few examples)

- Think of all the data that comes from cell phones, it's cheap to collect and there's a ton of it.
- Machines embedded with software/sensors (commonly referred to as the Internet of Things or IoT) produce a lot of data as well.

2. Computational Scale (NG MLY 01 pg 10)

The rise of the big data era has given us access to astounding amounts of data. That phenomenon paired with the exponential growth we've experienced in computational advances, has created the perfect storm for the emergence of the field of machine learning.

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

2018-09-05

Machine Learning From Scratch

└ Steps in the Machine Learning Process

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GRU Crash dataset

Now that we have some motivation for what machine learning is and what it can do, let's move into the steps of the machine learning process.



Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 0: Identify The Problem



Step 0 is to identify a problem that can be framed in such a way that it can be solved using machine learning.

Let's say that you work for the city of Grand Rapids, and you find that there are an increased number of hit and runs when the driver 1 was drinkin.

Do some research, maybe find a way to graph this?? You can do it!

Get the Data

This may look like:

- ▶ SQL query
- ▶ CSV download
- ▶ Web-scraping
- ▶ Designing experiments/surveys and collecting data yourself

Get the Data

This may look like:

- ▶ SQL query
- ▶ CSV download
- ▶ Web-scraping
- ▶ Designing experiments/surveys and collecting data yourself

In our case, we'll head to [GRData](#).

Machine Learning From Scratch

Steps in the Machine Learning Process

Step 1: Get the Data

Get the Data

[Get the Data](#)

This may look like:

- SQL query
- CSV download
- Web-scraping
- Designing experiments/surveys and collecting data yourself

In our case, we'll head to [GRData](#).

Step 1! Obtaining the data you'll need looks very different depending on what domain you're working in. In some instances, it can be fairly simple and straightforward, for example, In a business context, most often it will require querying some sort of internal database. Could also be downloading a csv file. In other instances, it may require a bit more creativity – For particular social research, you may need to scrape the web. In some cases, you may even need to collect some data yourself! Here are two examples:

1. You're developing a new data product at your company and are collecting data to fuel it
2. You're in public health and are working to make healthcare accessible to all residents of the greater GR area. You may need to conduct your own research to identify what may be inhibiting people from reaching healthcare.

In our case, we're lucky enough to have access to a meticulously maintained public database on the city of GR: GRData. Scroll through,

Get the Data

In our case, we'll head to [GRData](#)

```
In [33]: crash = pd.read_csv('Data/CGR_Crash_Data.csv')
         crash.head()
```

```
Out[33]:
```

	X	Y	OBJECTID	ROADSOFTID	BIKE	CITY	CRASHDATE	CRASHSEVER	CRASHTYPE	WORKZNEACT	...
0	-85.639647	42.927216	6001	929923	No	Grand Rapids	2007-02-16	Property Damage Only	Side-Swipe Same	Uncoded & Errors	...
1	-85.639487	42.927213	6002	935745	No	Grand Rapids	2007-06-22	Property Damage Only	Side-Swipe Same	Uncoded & Errors	...
2	-85.639387	42.927212	6003	926813	No	Grand Rapids	2007-01-08	Property Damage Only	Head-on	Work on Shoulder / Median	...
3	-85.639288	42.927210	6004	943813	No	Grand Rapids	2007-11-12	Property Damage Only	Side-Swipe Same	Uncoded & Errors	...
4	-85.639288	42.927210	6005	943791	No	Grand Rapids	2007-11-09	Property Damage Only	Parking	Uncoded & Errors	...

5 rows × 77 columns

- Machine Learning From Scratch
 - Steps in the Machine Learning Process
 - Step 1: Get the Data
 - Get the Data

In our case, we'll head to [GRData](#)

col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14	col15	col16	col17	col18	col19	col20	col21	col22	col23	col24	col25	col26	col27	col28	col29	col30	col31	col32	col33	col34	col35	col36	col37	col38	col39	col40	col41	col42	col43	col44	col45	col46	col47	col48	col49	col50	col51	col52	col53	col54	col55	col56	col57	col58	col59	col60	col61	col62	col63	col64	col65	col66	col67	col68	col69	col70	col71	col72	col73	col74	col75	col76	col77	col78	col79	col80	col81	col82	col83	col84	col85	col86	col87	col88	col89	col90	col91	col92	col93	col94	col95	col96	col97	col98	col99	col100	col101	col102	col103	col104	col105	col106	col107	col108	col109	col110	col111	col112	col113	col114	col115	col116	col117	col118	col119	col120	col121	col122	col123	col124	col125	col126	col127	col128	col129	col130	col131	col132	col133	col134	col135	col136	col137	col138	col139	col140	col141	col142	col143	col144	col145	col146	col147	col148	col149	col150	col151	col152	col153	col154	col155	col156	col157	col158	col159	col160	col161	col162	col163	col164	col165	col166	col167	col168	col169	col170	col171	col172	col173	col174	col175	col176	col177	col178	col179	col180	col181	col182	col183	col184	col185	col186	col187	col188	col189	col190	col191	col192	col193	col194	col195	col196	col197	col198	col199	col200	col201	col202	col203	col204	col205	col206	col207	col208	col209	col210	col211	col212	col213	col214	col215	col216	col217	col218	col219	col220	col221	col222	col223	col224	col225	col226	col227	col228	col229	col230	col231	col232	col233	col234	col235	col236	col237	col238	col239	col240	col241	col242	col243	col244	col245	col246	col247	col248	col249	col250	col251	col252	col253	col254	col255	col256	col257	col258	col259	col260	col261	col262	col263	col264	col265	col266	col267	col268	col269	col270	col271	col272	col273	col274	col275	col276	col277	col278	col279	col280	col281	col282	col283	col284	col285	col286	col287	col288	col289	col290	col291	col292	col293	col294	col295	col296	col297	col298	col299	col300	col301	col302	col303	col304	col305	col306	col307	col308	col309	col310	col311	col312	col313	col314	col315	col316	col317	col318	col319	col320	col321	col322	col323	col324	col325	col326	col327	col328	col329	col330	col331	col332	col333	col334	col335	col336	col337	col338	col339	col340	col341	col342	col343	col344	col345	col346	col347	col348	col349	col350	col351	col352	col353	col354	col355	col356	col357	col358	col359	col360	col361	col362	col363	col364	col365	col366	col367	col368	col369	col370	col371	col372	col373	col374	col375	col376	col377	col378	col379	col380	col381	col382	col383	col384	col385	col386	col387	col388	col389	col390	col391	col392	col393	col394	col395	col396	col397	col398	col399	col400	col401	col402	col403	col404	col405	col406	col407	col408	col409	col410	col411	col412	col413	col414	col415	col416	col417	col418	col419	col420	col421	col422	col423	col424	col425	col426	col427	col428	col429	col430	col431	col432	col433	col434	col435	col436	col437	col438	col439	col440	col441	col442	col443	col444	col445	col446	col447	col448	col449	col450	col451	col452	col453	col454	col455	col456	col457	col458	col459	col460	col461	col462	col463	col464	col465	col466	col467	col468	col469	col470	col471	col472	col473	col474	col475	col476	col477	col478	col479	col480	col481	col482	col483	col484	col485	col486	col487	col488	col489	col490	col491	col492	col493	col494	col495	col496	col497	col498	col499	col500	col501	col502	col503	col504	col505	col506	col507	col508	col509	col510	col511	col512	col513	col514	col515	col516	col517	col518	col519	col520	col521	col522	col523	col524	col525	col526	col527	col528	col529	col530	col531	col532	col533	col534	col535	col536	col537	col538	col539	col540	col541	col542	col543	col544	col545	col546	col547	col548	col549	col550	col551	col552	col553	col554	col555	col556	col557	col558	col559	col560	col561	col562	col563	col564	col565	col566	col567	col568	col569	col570	col571	col572	col573	col574	col575	col576	col577	col578	col579	col580	col581	col582	col583	col584	col585	col586	col587	col588	col589	col590	col591	col592	col593	col594	col595	col596	col597	col598	col599	col600	col601	col602	col603	col604	col605	col606	col607	col608	col609	col610	col611	col612	col613	col614	col615	col616	col617	col618	col619	col620	col621	col622	col623	col624	col625	col626	col627	col628	col629	col630	col631	col632	col633	col634	col635	col636	col637	col638	col639	col640	col641	col642	col643	col644	col645	col646	col647	col648	col649	col650	col651	col652	col653	col654	col655	col656	col657	col658	col659	col660	col661	col662	col663	col664	col665	col666	col667	col668	col669	col670	col671	col672	col673	col674	col675	col676	col677	col678	col679	col680	col681	col682	col683	col684	col685	col686	col687	col688	col689	col690	col691	col692	col693	col694	col695	col696	col697	col698	col699	col700	col701	col702	col703	col704	col705	col706	col707	col708	col709	col710	col711	col712	col713	col714	col715	col716	col717	col718	col719	col720	col721	col722	col723	col724	col725	col726	col727	col728	col729	col730	col731	col732	col733	col734	col735	col736	col737	col738	col739	col740	col741	col742	col743	col744	col745	col746	col747	col748	col749	col750	col751	col752	col753	col754	col755	col756	col757	col758	col759	col760	col761	col762	col763	col764	col765	col766	col767	col768	col769	col770	col771	col772	col773	col774	col775	col776	col777	col778	col779	col780	col781	col782	col783	col784	col785	col786	col787	col788	col789	col790	col791	col792	col793	col794	col795	col796	col797	col798	col799	col800	col801	col802	col803	col804	col805	col806	col807	col808	col809	col810	col811	col812	col813	col814	col815	col816	col817	col818	col819	col820	col821	col822	col823	col824	col825	col826	col827	col828	col829	col830	col831	col832	col833	col834	col835	col836	col837	col838	col839	col840	col841	col842	col843	col844	col845	col846	col847	col848	col849	col850	col851	col852	col853	col854	col855	col856	col857	col858	col859	col860	col861	col862	col863	col864	col865	col866	col867	col868	col869	col870	col871	col872	col873	col874	col875	col876	col877	col878	col879	col880	col881	col882	col883	col884	col885	col886	col887	col888	col889	col890	col891	col892	col893	col894	col895	col896	col897	col898	col899	col900	col901	col902	col903	col904	col905	col906	col907	col908	col909	col910	col911	col912	col913	col914	col915	col916	col917	col918	col919	col920	col921	col922	col923	col924	col925	col926	col927	col928	col929	col930	col931	col932	col933	col934	col935	col936	col937	col938	col939	col940	col941	col942	col943	col944	col945	col946	col947	col948	col949	col950	col951	col952	col953	col954	col955	col956	col957	col958	col959	col960	col961	col962	col963	col964	col965	col966	col967	col968	col969	col970	col971	col972	col973	col974	col975	col976	col977	col978	col979	col980	col981	col982	col983	col984	col985	col986	col987	col988	col989	col990	col991	col992	col993	col994	col995	col996	col997	col998	col999	col1000
col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14	col15	col16	col17	col18	col19	col20	col21	col22	col23	col24	col25	col26	col27	col28	col29	col30	col31	col32	col33	col34	col35	col36	col37	col38	col39	col40	col41	col42	col43	col44	col45	col46	col47	col48	col49	col50	col51	col52	col53	col54	col55	col56	col57	col58	col59	col60	col61	col62	col63	col64	col65	col66	col67	col68	col69	col70	col71	col72	col73	col74	col75	col76	col77	col78	col79	col80	col81	col82	col83	col84	col85	col86	col87	col88	col89	col90	col91	col92	col93	col94	col95	col96	col97	col98	col99	col100	col101	col102	col103	col104	col105	col106	col107	col108	col109	col110	col111	col112	col113	col114	col115	col116	col117	col118	col119	col120	col121	col122	col123	col124	col125	col126	col127	col128	col129	col130	col131	col132	col133	col134	col135	col136	col137	col138	col139	col140	col141	col142	col143	col144	col145	col146	col147	col148	col149	col150	col151	col152	col153	col154	col155	col156	col157	col158	col159	col160	col161	col162	col163	col164	col165	col166	col167	col168	col169	col170	col171	col172	col173	col174	col175	col176	col177	col178	col179	col180	col181	col182	col183	col184	col185	col186	col187	col188	col189	col190	col191	col192	col193	col194	col195	col196	col197	col198	col199	col200	col201	col202	col203	col204	col205	col206	col207	col208	col209	col210	col211	col212	col213	col214	col215	col216	col217	col218	col219	col220	col221	col222	col223	col224	col225	col226	col227	col228	col229	col230	col231	col232	col233	col234	col235	col236	col237	col238	col239	col240	col241	col242	col243	col244	col245	col246	col247	col248	col249	col250	col251	col252	col253	col254	col255	col256	col257	col258	col259	col260	col261	col262	col263	col264	col265	col266	col267	col268	col269	col270	col271	col272	col273	col274	col275	col276	col277	col278	col279	col280	col281	col282	col283	col284	col285	col286	col287	col288	col289	col290	col291	col292	col293	col294	col295	col296	col297	col298	col299	col300	col301	col302	col303	col304	col305	col306	col307	col308	col309	col310	col311	col312	col313	col314	col315	col316	col317	col318	col319	col320	col321	col322	col323	col324	col325	col326	col327	col328	col329	col330	col331	col332	col333	col334	col335	col336	col337	col338	col339	col340	col341	col342	col343	col344	col345	col346	col347	col348	col349	col350	col351	col352	col353	col354	col355	col356	col357	col358	col359	col360	col361	col362	col363	col364	col365	col366	col367	col368	col369	col370	col371	col372	col373	col374	col375	col376	col377	col378	col379	col380	col381	col382	col383	col384	col385	col386	col387	col388	col389	col390	col391	col392	col393	col394	col395	col396	col397	col398	col399																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									

* note something about how we'll often refer to each row as an observation and each column as a feature

Explore the Data

- ▶ Verify data
- ▶ Visualize data
- ▶ Identify patterns
- ▶ Give direction to analysis

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 2: Data Exploration

└ Explore the Data

Explore the Data

- Verify data
- Visualize data
- Identify patterns
- Give direction to analysis

Step 2 is to explore the data! This is kind of an unstructured approach to understanding initial patterns in the data and potentially points of interest. This process isn't meant to reveal every bit of information a dataset holds, but rather give you direction in your analysis and potentially give you clues in how to process/model the data.[1] Now, if you're just emailed a csv file, this step is especially crucial, and it may take you some time to explore the data, get a feeling for what you're dealing with. If you are analyzing data that you work with day in and day out, this "exploration" process may be a bit more implicit.

The main idea here is to build an understanding of your data. Without an appreciation for the context of the data, it's just numbers. But when you see the data in context, it's fascinating, it's a story.

More often than not, your exploration of the data leads to more questions than answers.

Machine Learning From Scratch

└ Steps in the Machine Learning Process

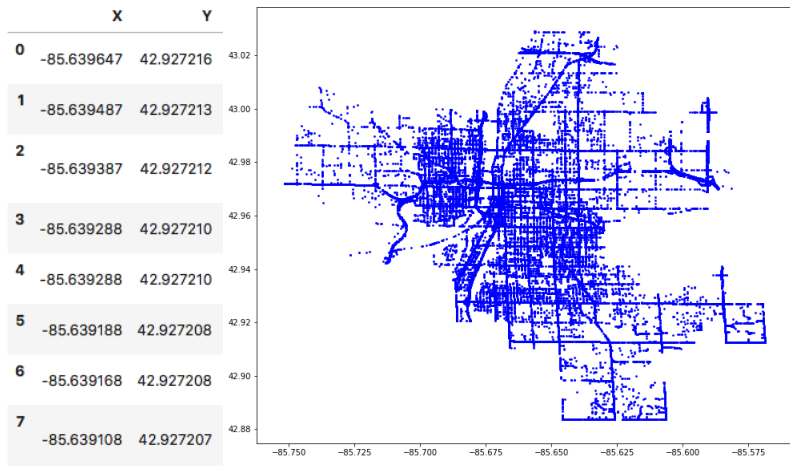
└ Step 2: Data Exploration

	X	Y
0	-85.639647	42.927216
1	-85.639487	42.927213
2	-85.639387	42.927212
3	-85.639288	42.927210
4	-85.639288	42.927210
5	-85.639188	42.927208
6	-85.639168	42.927208
7	-85.639108	42.927207

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 2: Data Exploration

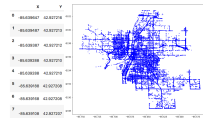


2018-09-05

Machine Learning From Scratch

Steps in the Machine Learning Process

Step 2: Data Exploration

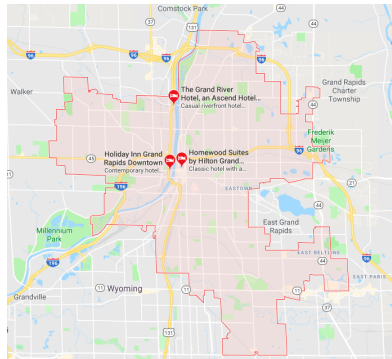
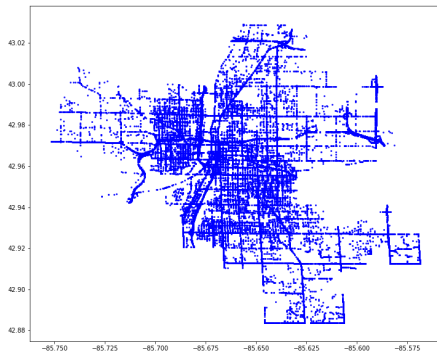


With our dataset, on car crashes, a logical place to begin would be the first two columns, containing latitudes and longitudes of each crash. This is just a snapshot of the data on the left side, and on the right, each dot represents a car crash.

Machine Learning From Scratch

Steps in the Machine Learning Process

Step 2: Data Exploration



2018-09-05

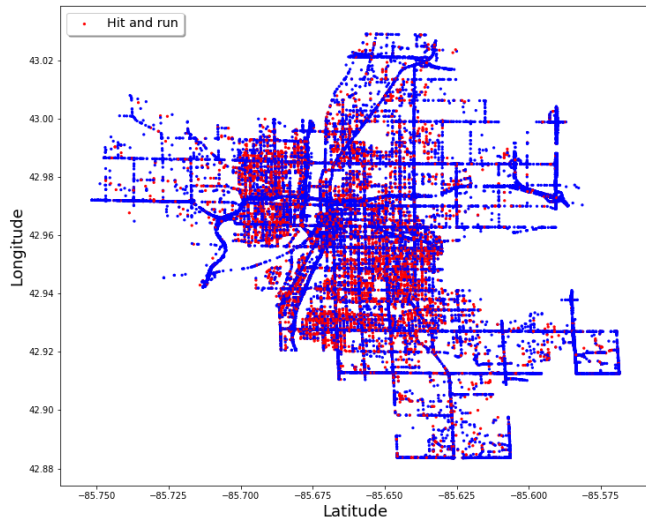
Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 2: Data Exploration



Now, this is pretty telling about our data, remember, there is nearly 73,000 crashes recorded, and if we juxtapose this plot with the city of GR, we actually see that the plot of crashes outline the city boundaries!

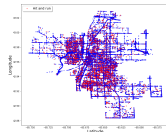


2018-09-05

Machine Learning From Scratch

Steps in the Machine Learning Process

Step 2: Data Exploration



We may also be interested in hit and runs...

This plot again shows each car crash, but this time with each red dot representing a hit and run

Check the variable's distribution

```
In [34]: crash = pd.read_csv('Data/CGR_Crash_Data.csv')
         crash.head(3)
```

```
Out[34]:
```

	X	Y	OBJECTID	ROADSOFTID	BIKE	CITY	CRASHDATE	CRASHSEVER	CRASHTYPE	WORKZNEACT	...
0	-85.639647	42.927216	6001	929923	No	Grand Rapids	2007-02-16	Property Damage Only	Side-Swipe Same	Uncoded & Errors	...
1	-85.639487	42.927213	6002	935745	No	Grand Rapids	2007-06-22	Property Damage Only	Side-Swipe Same	Uncoded & Errors	...
2	-85.639387	42.927212	6003	926813	No	Grand Rapids	2007-01-08	Property Damage Only	Head-on	Work on Shoulder / Median	...

3 rows x 77 columns

```
In [32]: crash.VEH3TYPE.value_counts()
```

```
Out[32]:
```

Uncoded & Errors	67212
Passenger Car, SUV, Van	4788
Pickup Truck	503
Motorhome	327
Truck Under 10,000 lbs	63
Truck / Bus (Commercial)	62
Other Non-Commercial	10
Motorcycle	10
Go-cart / Golf Cart	2

Name: VEH3TYPE, dtype: int64

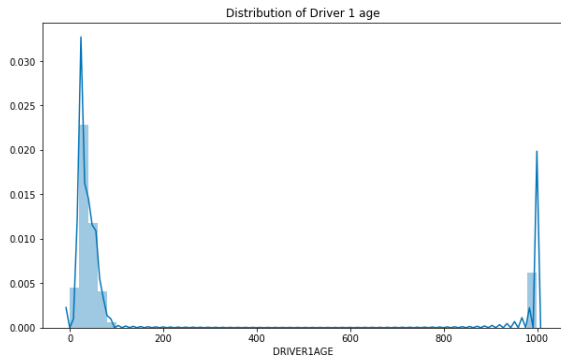
[illegible]

When we check the counts of the different values found in the "VEH3" column, we see that over 67,000 of them are errors! Now, this doesn't mean that the column is useless, but in terms of building a predictive model, this column probably won't be much help, so as we'll see in the data processing step, we'll end up dropping it.

Check the variable's distribution

```
In [41]: fig, ax = plt.subplots(figsize=(10,6))  
         ax.set_title('Distribution of Driver 1 age')  
         sns.distplot(crash.DRIVER1AGE)
```

```
Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1a742080>
```

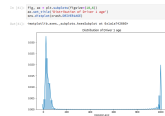


2018-09-05

Machine Learning From Scratch

- Steps in the Machine Learning Process
 - Step 2: Data Exploration
 - Check the variable's distribution

Check the variable's distribution

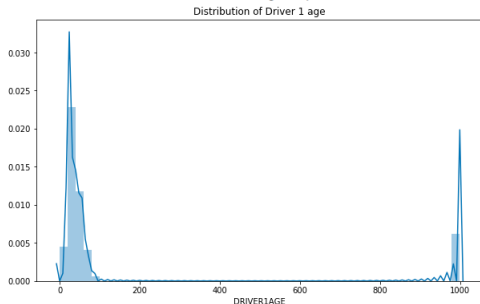


Continuing on, here we check the distribution of the age's of all the "DRIVER1"s recorded in the dataset. In my head, I would think that age would be an important feature in car crashes. And so we check it out, and it appears that there are a ton of instances bunched up between 1 and 100... that makes sense... then there is also a decent cluster of observations around 1000 years old... that does not make sense.

Check the variable's distribution

```
In [45]: fig, ax = plt.subplots(figsize=(10,6))  
         ax.set_title('Distribution of Driver 1 age')  
         sns.distplot(crash.DRIVER1AGE)
```

There are 8979 Driver 1's recorded as being 999 years-old.



```
In [46]: print('There are', crash.DRIVER1AGE[crash.DRIVER1AGE == 999].count(), "driver 1's recorded as being 999 years-old.")
```

There are 8979 driver 1's recorded as being 999 years-old.

Machine Learning From Scratch

- └ Steps in the Machine Learning Process
 - └ Step 2: Data Exploration
 - └ Check the variable's distribution

Check the variable's distribution



Nearly 9,000 driver 1's are recorded as being 999 years-old.. That is a good thing to know before trying to build a predictive model, as it seriously skews the data. We'll address that in our data processing stage.

Next, we move onto data preparation. This is the stage where we make the final manipulations to our data before feeding it into our ML algorithm. Now, you could make an argument that preparing the data is the most important part of the machine learning workflow. After all it's the data that fuels the algorithm, garbage in, garbage out! It's been shown time and time again that more/bigger data beats a better algorithm everytime. Though it usually appears to be straightforward, this step can often require a lot of creativity.

Data Selection

Use as minimal features as possible

1. Computationally efficient
2. Easier to interpret
3. Simpler is better

Data Selection

Use as minimal features as possible

1. Computationally efficient
2. Easier to interpret
3. Simpler is better

```
In [8]: crash = pd.read_csv('Data/CGR_Crash_Data.csv')
crash = crash[['X', 'Y', 'CRASHSEVER', 'DRIVER1AGE', 'DRIVER1SEX',
               'EMRGVEH', 'HITANDRUN', 'SPEEDLIMIT', 'HOUR', 'MOTORCYCLE',
               'NUMOFINJ', 'D1COND', 'D1DRINKIN']]
crash.columns

Out[8]: Index(['X', 'Y', 'CRASHSEVER', 'DRIVER1AGE', 'DRIVER1SEX', 'EMRGVEH',
              'HITANDRUN', 'SPEEDLIMIT', 'HOUR', 'MOTORCYCLE', 'NUMOFINJ', 'D1COND',
              'D1DRINKIN'],
              dtype='object')
```

1. Computationally efficient
2. Easier to interpret
3. Simpler is better

This kicks off step 3, Data Preparation! The first part in preparing the data is simply choosing which features (you can think of as columns in the spreadsheet) you'll want to use in your model. As seen in our EDA some columns have a lot of missing data, and we'll drop them entirely. It's generally regarded as best practice to use as minimal amount of features as possible, such that your predictive model still predicts as accurately as you need it to.

- * computationally efficient
- * more easily interpretable
- * simpler is better

So how do we actually determine which features to use?, we can use various statistical tests, and even some algorithms to determine which features are going to be most relevant to predicting our target variable (which for us is whether or not the driver who caused the crash was drinking). We won't go into detail here.

When you are first beginning to iterate through different ml models, it is okay to kind of eyeball it or use what you know about the context of the

Feature Engineering



'Coming up with features is difficult, time-consuming, requires expert knowledge. Applied machine learning is basically feature engineering.'

Prof. Andrew Ng



'At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.'

Prof. Pedro Domingos

Machine Learning From Scratch

Steps in the Machine Learning Process

Step 3: Data Preparation

Feature Engineering

Feature Engineering



"Coming up with features is difficult, time-consuming, requires expert knowledge. Applied machine learning is basically feature engineering."
Prof. Andrew Ng



"At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used."
Prof. Pedro Domingos

So now we have our 13 features that we've chosen to use, and we could just send these raw features into our algorithm, and it may perform well, but it may not. It's important to remember that our ultimate goal is to build a predictive model that can make accurate predictions on new/unseen observations. When all the data comes in from a car crash that just happened, we want to be able to accurately predict whether or not it was a drunk driver that caused it.

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data. - Jason Brownlee [2]

We acknowledge, however, that the data that's being recorded (from those UD10 reports) isn't necessarily guaranteed to accurately represent reality. Which is an important thing if we want to build accurate, stable predictive models.

The examples that we show here are absolutely not exhaustive when it

Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 12 \\ 2 \\ 4 \\ 18 \\ 19 \\ 6 \\ \vdots \end{bmatrix}$$

Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 12 \\ 2 \\ 4 \\ 18 \\ 19 \\ 6 \\ \vdots \end{bmatrix} \implies f(x) = \frac{2 \cdot \pi \cdot (\text{hour})}{24}$$

Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 12 \\ 2 \\ 4 \\ 18 \\ 19 \\ 6 \\ \vdots \end{bmatrix} \Rightarrow f(x) = \frac{2 \cdot \pi \cdot (\text{hour})}{24} \Rightarrow \begin{bmatrix} 3.14 \\ 0.52 \\ 1.03 \\ 4.71 \\ 4.98 \\ 1.57 \\ \vdots \end{bmatrix}$$

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

└ Feature Engineering: *transforming "hour" variable*Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 12 \\ 2 \\ 4 \\ 18 \\ 19 \\ 6 \\ 5 \\ \vdots \end{bmatrix} \implies f(x) = \frac{2 - \sin(\frac{\text{hour}}{24})}{24} \implies \begin{bmatrix} 3.14 \\ 0.52 \\ 1.03 \\ 4.71 \\ 4.98 \\ 1.57 \\ \vdots \end{bmatrix}$$

For example, one of our columns contains the hour the crash occurred at (ranging from 0-23). It did not make sense to me to treat this column as a numerical variable, since the model would interpret 2pm as twice 1pm... while that does not make logical sense. However, I felt that to treat each hour as its own category wouldn't accurately capture the information that the hour conveys (for example, it would not capture the fact that 12pm and 12am are 12 hours apart, while 12pm and 1pm are next to each other), so I set out to transform this feature in order to convey more meaning to our algorithm. Here's what I did.

On the left we have a vector containing a few of the hours that the crashes took place at. I sent this vector through this trigonometric function that we have here.

Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 3.14 \\ 0.52 \\ 1.03 \\ 4.71 \\ 4.98 \\ 1.57 \\ \vdots \end{bmatrix} \Rightarrow \underbrace{\begin{bmatrix} 0.00 \\ 0.47 \\ 0.86 \\ -0.99 \\ -0.97 \\ 1.0 \\ \vdots \end{bmatrix}}_{\sin(f(x))}, \underbrace{\begin{bmatrix} -1.0 \\ 0.87 \\ 0.51 \\ -0.002 \\ -0.26 \\ 0.001 \\ \vdots \end{bmatrix}}_{\cos(f(x))}$$

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

└ Feature Engineering: *transforming "hour" variable*Feature Engineering: *transforming "hour" variable*

$$\begin{bmatrix} 3.14 \\ 0.52 \\ 1.03 \\ 4.71 \\ 4.98 \\ 1.57 \\ \vdots \end{bmatrix} \implies \begin{bmatrix} 0.00 \\ 0.47 \\ 0.86 \\ -0.99 \\ -0.97 \\ 1.0 \\ \vdots \end{bmatrix}, \begin{bmatrix} -1.0 \\ 0.51 \\ -0.002 \\ -0.26 \\ 0.001 \\ \vdots \end{bmatrix}$$

$\sin^2(x)$ $\cos^2(x)$

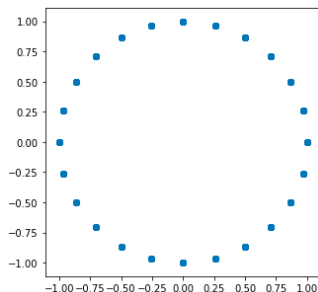
Okay, so we have this new vector, but it doesn't look too helpful at the moment. well, we take that output and map it to two separate vectors, a sin and cos transformed vector.

Feature Engineering: *transforming "hour" variable*

```
In [193]: crash['HOUR_X']=np.sin(2. * np.pi * crash.HOUR / 24.)  
crash['HOUR_Y']=np.cos(2. * np.pi * crash.HOUR / 24.)
```

```
In [194]: # Hence, the time of day is now cyclic (just as in reality)  
plt.figure(figsize = (5,5))  
plt.scatter(crash.HOUR_X, crash.HOUR_Y)
```

```
Out[194]: <matplotlib.collections.PathCollection at 0x1a0daeca20>
```



Machine Learning From Scratch

Steps in the Machine Learning Process

Step 3: Data Preparation

Feature Engineering: *transforming "hour" variable*

Feature Engineering: *transforming "hour" variable*

```
In [109]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

In [109]: # Create the time of day for new vehicle based on the meeting
plt.figure(figsize=(10,10))
plt.scatter(hour_data['h'], hour_data['s'])
```



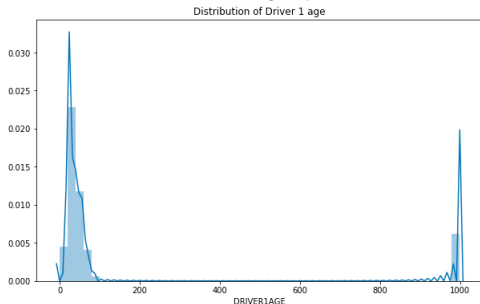
So.. why did we do all of that? Well, when we transfer that all into code, and graph the two vectors on the x-y plane, we see that they make a perfect circle... kind of like a clock..

This is exactly what we want. We've mapped our single hour column to two separate hour columns (think of them like x and y coordinates for each hour). This cyclic representation of time conveys the hour of the car crash in a way that has meaning to the algorithm. This is a really important aspect of feature engineering.

Feature Engineering: *imputing missing ages*

```
In [45]: fig, ax = plt.subplots(figsize=(10,6))  
ax.set_title('Distribution of Driver 1 age')  
sns.distplot(crash.DRIVER1AGE)
```

There are 8979 Driver 1's recorded as being 999 years-old.



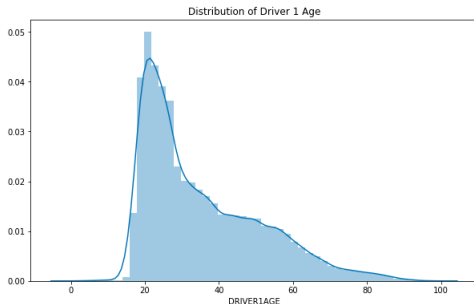
```
In [46]: print('There are', crash.DRIVER1AGE[crash.DRIVER1AGE == 999].count(), "driver 1's recorded as being 999 years-old.")
```

There are 8979 driver 1's recorded as being 999 years-old.

Feature Engineering: *imputing missing ages*

```
In [90]: # This age distribution looks much better!  
fig, ax = plt.subplots(figsize=(10,6))  
ax.set_title('Distribution of Driver 1 Age')  
sns.distplot(crash['DRIVER1AGE'])
```

```
Out[90]: <matplotlib.axes._subplots.AxesSubplot at 0x1a21b50240>
```



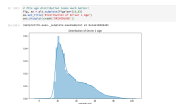
Machine Learning From Scratch

Steps in the Machine Learning Process

Step 3: Data Preparation

Feature Engineering: *imputing missing ages*

Feature Engineering: *imputing missing ages*



Now, earlier when we were exploring our data and checking the distribution of the ages, we found that there are about 9,000 erroneous ages.

When you have erroneous/outright missing data you have a couple different options.

1. drop them entirely
2. impute mean/median (make this choice depending on distribution of data)

For most cases, one of those two options will work just fine. In our case, with nearly 9,000 errors, I'm not sure any of those would make sense, so we'll opt for a bit more involved technique. (The technical term is "Multiple Imputation by Chained Equations", affectionately known as MICE). What this means is that we will fit a linear regression model to our data to predict what the missing ages could be. We can think of these as making an educated guess.

Data Processing

Two general types of data to deal with:

- ▶ Numerical variables (Quantitative)
 - ▶ Driver 1 age, number of injuries, etc
- ▶ Categorical variables (Qualitative)
 - ▶ Hit and run, motorcycle involved, etc

Machine Learning From Scratch

Steps in the Machine Learning Process

Step 3: Data Preparation

Data Processing

Data Processing

Two general types of data to deal with:

- Numerical variables (Quantitative)
 - Driver 1 age, number of injuries, etc
- Categorical variables (Qualitative)
 - Hit and run, motorcycle involved, etc

Now, we move onto processing the data that we've selected. The data processing stage naturally diverges into two substeps: dealing with numerical variables, and dealing with categorical variables.

It's important to note that some of the preprocessing steps we'll talk about here may actually be necessary for you to do to get the data in a format where you can explore it.

Numerical variables are quantitative – it's something you can measure. In our case, some numerical variables are Driver1 age, and number of injuries.

Categorical variables are qualitative, in our case, we have some binary categorical variables: like whether or not the crash was a hit and run, whether or not a motorcycle was involved. It's either one or the other. Variables can of course have multiple categories, for example, which car insurance provider you choose: (Nationwide, Statefarm, BlueCross BlueShield) There is obviously a lot of them, but all drivers fit into one of those categories... or at least they should.

There is some grey area between the two – maybe mention speedlimits or

Data Processing: *numerical variables*

```
In [91]: crash[['X', 'Y', 'DRIVER1AGE', 'NUMOFINJ']].head()
```

```
Out[91]:
```

	X	Y	DRIVER1AGE	NUMOFINJ
0	-85.639647	42.927216	62.0	0
1	-85.639487	42.927213	31.0	0
2	-85.639387	42.927212	22.0	0
3	-85.639288	42.927210	30.0	0
4	-85.639288	42.927210	44.0	0

Data Processing: *numerical variables*

```
In [34]: sc_X = StandardScaler()
X_scaled = sc_X.fit_transform(crash[['X', 'Y', 'DRIVER1AGE', 'NUMOFINJ']])
X_scaled_df = pd.DataFrame(X_scaled, columns = ['X', 'Y', 'DRIVER1AGE', 'NUMOFINJ'])
crash_ = crash.drop(['X', 'Y', 'DRIVER1AGE', 'NUMOFINJ'], axis=1)
crash = pd.concat([X_scaled_df, crash_], axis=1)
crash.iloc[:, :4].head()
```

```
Out[34]:
```

	X	Y	DRIVER1AGE	NUMOFINJ
0	0.406318	-0.996140	1.685157	-0.416816
1	0.411006	-0.996237	-0.269018	-0.416816
2	0.413936	-0.996298	-0.836360	-0.416816
3	0.416866	-0.996358	-0.332056	-0.416816
4	0.416866	-0.996358	0.550474	-0.416816

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

└ Data Processing: *numerical variables*Data Processing: *numerical variables*

```

10 [10]: m2 = RandomForestRegressor()
11         X_scaled = StandardScaler().fit(X).transform(X)
12         y_scaled = StandardScaler().fit(y).transform(y)
13         m2.fit(X_scaled, y_scaled)
14         y_pred = m2.predict(X_scaled)
15         y_pred = StandardScaler().inverse_transform(y_pred)
16         print('R^2: %.3f' % r2)
17         print('MSE: %.3f' % mse)
18         print('RMSE: %.3f' % rmse)
19         print('MAE: %.3f' % mae)
20         print('Max Error: %.3f' % max_error)
21         print('Mean Error: %.3f' % mean_error)
22         print('Mean Absolute Error: %.3f' % mae)
23         print('Mean Squared Error: %.3f' % mse)
24         print('Root Mean Squared Error: %.3f' % rmse)
25         print('Coefficient of Determination: %.3f' % r2)
26         print('Adjusted Coefficient of Determination: %.3f' % adj_r2)
27         print('F-statistic: %.3f' % f_stat)
28         print('P-value: %.3f' % p_value)
29         print('t-statistic: %.3f' % t_stat)
30         print('Standard Error: %.3f' % std_err)
31         print('Variance Inflation Factor: %.3f' % vif)
32         print('Durbin-Watson Statistic: %.3f' % dw_stat)
33         print('Akaike Information Criterion: %.3f' % aic)
34         print('Bayesian Information Criterion: %.3f' % bic)
35         print('Hannan-Quinn Criterion: %.3f' % hqic)
36         print('Schwarz Criterion: %.3f' % scic)
37         print('Consistent Akaike Information Criterion: %.3f' % caic)
38         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
39         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
40         print('Consistent Schwarz Criterion: %.3f' % cscic)
41         print('Consistent Akaike Information Criterion: %.3f' % caic)
42         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
43         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
44         print('Consistent Schwarz Criterion: %.3f' % cscic)
45         print('Consistent Akaike Information Criterion: %.3f' % caic)
46         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
47         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
48         print('Consistent Schwarz Criterion: %.3f' % cscic)
49         print('Consistent Akaike Information Criterion: %.3f' % caic)
50         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
51         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
52         print('Consistent Schwarz Criterion: %.3f' % cscic)
53         print('Consistent Akaike Information Criterion: %.3f' % caic)
54         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
55         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
56         print('Consistent Schwarz Criterion: %.3f' % cscic)
57         print('Consistent Akaike Information Criterion: %.3f' % caic)
58         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
59         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
60         print('Consistent Schwarz Criterion: %.3f' % cscic)
61         print('Consistent Akaike Information Criterion: %.3f' % caic)
62         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
63         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
64         print('Consistent Schwarz Criterion: %.3f' % cscic)
65         print('Consistent Akaike Information Criterion: %.3f' % caic)
66         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
67         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
68         print('Consistent Schwarz Criterion: %.3f' % cscic)
69         print('Consistent Akaike Information Criterion: %.3f' % caic)
70         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
71         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
72         print('Consistent Schwarz Criterion: %.3f' % cscic)
73         print('Consistent Akaike Information Criterion: %.3f' % caic)
74         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
75         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
76         print('Consistent Schwarz Criterion: %.3f' % cscic)
77         print('Consistent Akaike Information Criterion: %.3f' % caic)
78         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
79         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
80         print('Consistent Schwarz Criterion: %.3f' % cscic)
81         print('Consistent Akaike Information Criterion: %.3f' % caic)
82         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
83         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
84         print('Consistent Schwarz Criterion: %.3f' % cscic)
85         print('Consistent Akaike Information Criterion: %.3f' % caic)
86         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
87         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
88         print('Consistent Schwarz Criterion: %.3f' % cscic)
89         print('Consistent Akaike Information Criterion: %.3f' % caic)
90         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
91         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
92         print('Consistent Schwarz Criterion: %.3f' % cscic)
93         print('Consistent Akaike Information Criterion: %.3f' % caic)
94         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
95         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
96         print('Consistent Schwarz Criterion: %.3f' % cscic)
97         print('Consistent Akaike Information Criterion: %.3f' % caic)
98         print('Consistent Bayesian Information Criterion: %.3f' % cbic)
99         print('Consistent Hannan-Quinn Criterion: %.3f' % chqic)
100        print('Consistent Schwarz Criterion: %.3f' % cscic)

```

Here, we have our numerical variables. One problem that we still face with these numerical variables is that when we feed our data through the algorithm, the computer will view an increase in 1 latitude of latitude as equivalent to an increase in 1 year of Age of the driver. In actuality and increase in 1 degree of latitude could land you in an entirely different neighborhood, while 37 y/o and 38 y/o are basically exactly the same. We will rescale the numerical variables through a process called standardization. This is not necessary for all ml algorithms, but generally will not hurt if we do it.

When we standardize the variables, we rescale them so that they all have a mean of 0 and a S.D. of 1.

Notice that in this snippet, the number of injuries was 0 for all of them, and now is negative... this means that having 0 injuries in a car crash is below average...

Data Processing: *categorical variables*

```
In [111]: crash[['CRASHSEVER', 'DRIVER1SEX', 'EMRGVEH', 'HITANDRUN',  
                'MOTORCYCLE', 'D1COND', 'D1DRINKIN']].head()
```

```
Out[111]:
```

	CRASHSEVER	DRIVER1SEX	EMRGVEH	HITANDRUN	MOTORCYCLE	D1COND	D1DRINKIN
0	Property Damage Only	F	No	Yes	No	Appeared Normal	No
1	Property Damage Only	M	No	Yes	No	Unknown	No
2	Property Damage Only	F	No	No	No	Appeared Normal	No
3	Property Damage Only	M	No	Yes	No	Appeared Normal	No
4	Property Damage Only	M	No	No	No	Appeared Normal	No

Data Processing: *categorical variables*

In [135]: `dummies.head()`

Out[135]:

	CRASHSEVER_Fatal	CRASHSEVER_Injury	CRASHSEVER_Property Damage Only	DRIVER1SEX_F	DRIVER1SEX_M	DRIVER1SEX_U
0	0	0	1	1	0	0
1	0	0	1	0	1	0
2	0	0	1	1	0	0
3	0	0	1	0	1	0
4	0	0	1	0	1	0

5 rows × 21 columns

Machine Learning From Scratch

└ Steps in the Machine Learning Process

└ Step 3: Data Preparation

└ Data Processing: *categorical variables*Data Processing: *categorical variables*

IN [107]: [display\(mtcars\)](#)

Out[107]:

	mpg	wt	qsec	vs	am	gear	carb
1	16.0	3.57	16.99	0	1	4	6
2	17.8	3.57	16.99	0	1	4	6
3	15.8	3.57	16.99	0	1	4	6
4	18.1	3.57	16.99	0	1	4	6
5	14.5	3.57	16.99	0	1	4	6
6	15.2	3.57	16.99	0	1	4	6
7	15.7	3.57	16.99	0	1	4	6
8	15.8	3.57	16.99	0	1	4	6
9	15.2	3.57	16.99	0	1	4	6
10	14.7	3.57	16.99	0	1	4	6
11	15.0	3.57	16.99	0	1	4	6
12	15.2	3.57	16.99	0	1	4	6
13	15.2	3.57	16.99	0	1	4	6
14	15.2	3.57	16.99	0	1	4	6
15	15.2	3.57	16.99	0	1	4	6
16	15.2	3.57	16.99	0	1	4	6
17	15.2	3.57	16.99	0	1	4	6
18	15.2	3.57	16.99	0	1	4	6
19	15.2	3.57	16.99	0	1	4	6
20	15.2	3.57	16.99	0	1	4	6
21	15.2	3.57	16.99	0	1	4	6
22	15.2	3.57	16.99	0	1	4	6
23	15.2	3.57	16.99	0	1	4	6
24	15.2	3.57	16.99	0	1	4	6
25	15.2	3.57	16.99	0	1	4	6
26	15.2	3.57	16.99	0	1	4	6
27	15.2	3.57	16.99	0	1	4	6
28	15.2	3.57	16.99	0	1	4	6
29	15.2	3.57	16.99	0	1	4	6
30	15.2	3.57	16.99	0	1	4	6
31	15.2	3.57	16.99	0	1	4	6
32	15.2	3.57	16.99	0	1	4	6
33	15.2	3.57	16.99	0	1	4	6
34	15.2	3.57	16.99	0	1	4	6
35	15.2	3.57	16.99	0	1	4	6
36	15.2	3.57	16.99	0	1	4	6
37	15.2	3.57	16.99	0	1	4	6
38	15.2	3.57	16.99	0	1	4	6
39	15.2	3.57	16.99	0	1	4	6
40	15.2	3.57	16.99	0	1	4	6
41	15.2	3.57	16.99	0	1	4	6
42	15.2	3.57	16.99	0	1	4	6
43	15.2	3.57	16.99	0	1	4	6
44	15.2	3.57	16.99	0	1	4	6
45	15.2	3.57	16.99	0	1	4	6
46	15.2	3.57	16.99	0	1	4	6
47	15.2	3.57	16.99	0	1	4	6
48	15.2	3.57	16.99	0	1	4	6
49	15.2	3.57	16.99	0	1	4	6
50	15.2	3.57	16.99	0	1	4	6
51	15.2	3.57	16.99	0	1	4	6
52	15.2	3.57	16.99	0	1	4	6
53	15.2	3.57	16.99	0	1	4	6
54	15.2	3.57	16.99	0	1	4	6
55	15.2	3.57	16.99	0	1	4	6
56	15.2	3.57	16.99	0	1	4	6
57	15.2	3.57	16.99	0	1	4	6
58	15.2	3.57	16.99	0	1	4	6
59	15.2	3.57	16.99	0	1	4	6
60	15.2	3.57	16.99	0	1	4	6
61	15.2	3.57	16.99	0	1	4	6
62	15.2	3.57	16.99	0	1	4	6
63	15.2	3.57	16.99	0	1	4	6
64	15.2	3.57	16.99	0	1	4	6
65	15.2	3.57	16.99	0	1	4	6
66	15.2	3.57	16.99	0	1	4	6
67	15.2	3.57	16.99	0	1	4	6
68	15.2	3.57	16.99	0	1	4	6
69	15.2	3.57	16.99	0	1	4	6
70	15.2	3.57	16.99	0	1	4	6
71	15.2	3.57	16.99	0	1	4	6
72	15.2	3.57	16.99	0	1	4	6
73	15.2	3.57	16.99	0	1	4	6
74	15.2	3.57	16.99	0	1	4	6
75	15.2	3.57	16.99	0	1	4	6
76	15.2	3.57	16.99	0	1	4	6
77	15.2	3.57	16.99	0	1	4	6
78	15.2	3.57	16.99	0	1	4	6
79	15.2	3.57	16.99	0	1	4	6
80	15.2	3.57	16.99	0	1	4	6
81	15.2	3.57	16.99	0	1	4	6
82	15.2	3.57	16.99	0	1	4	6
83	15.2	3.57	16.99	0	1	4	6
84	15.2	3.57	16.99	0	1	4	6
85	15.2	3.57	16.99	0	1	4	6
86	15.2	3.57	16.99	0	1	4	6
87	15.2	3.57	16.99	0	1	4	6
88	15.2	3.57	16.99	0	1	4	6
89	15.2	3.57	16.99	0	1	4	6
90	15.2	3.57	16.99	0	1	4	6
91	15.2	3.57	16.99	0	1	4	6
92	15.2	3.57	16.99	0	1	4	6
93	15.2	3.57	16.99	0	1	4	6
94	15.2	3.57	16.99	0	1	4	6
95	15.2	3.57	16.99	0	1	4	6
96	15.2	3.57	16.99	0	1	4	6
97	15.2	3.57	16.99	0	1	4	6
98	15.2	3.57	16.99	0	1	4	6
99	15.2	3.57	16.99	0	1	4	6
100	15.2	3.57	16.99	0	1	4	6

7 rows x 8 columns

Here we have our categorical variables. In our excel spreadsheet, this looks great, nice and clean.., but if we want to squeeze this data into a model, we need to manipulate it into a format that makes sense for math. Basically, we are going to turn all of our categories into 1's and 0's, representing yes' and no's. You can think of this as transforming the data to turn everything into a yes or no question.

Was the driver sex male? No. Was the driver sex female? yes.

I've seen this called creating dummy variables, or one-hot-encoding

This is just a snapshot, you see that our columns of categorical variables grew from 7 to 21

Come up with more concrete explanation

Choosing a Model/Representation

	Classification	Regression
Supervised	<ul style="list-style-type: none">• Logistic Regression• Naive-Bayes• KNN• SVM	<ul style="list-style-type: none">• Linear Regression• Decision Trees• Random Forests
Unsupervised	<ul style="list-style-type: none">• Apriori• Hidden Markov Model	<ul style="list-style-type: none">• PCA• K-means• SVD

Machine Learning From Scratch

Steps in the Machine Learning Process

Step 4: Model Selection

Choosing a Model/Representation

Choosing a Model/Representation

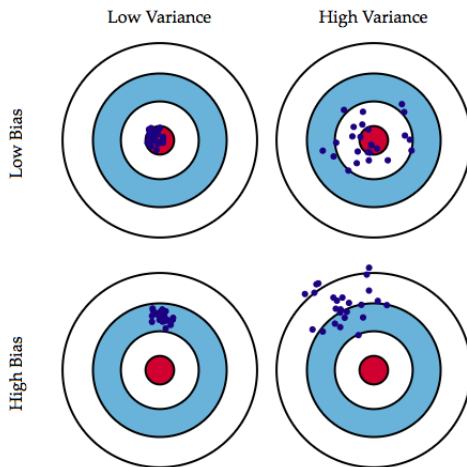
	Classification	Regression
Supervised	<ul style="list-style-type: none">• Logistic Regression• Naïve-Bayes• KNN• SVM	<ul style="list-style-type: none">• Linear Regression• Decision Trees• Random Forests
Unsupervised	<ul style="list-style-type: none">• Apriori• Hidden Markov Model	<ul style="list-style-type: none">• PCA• K-means• SVD

And with that, our data is ready to be fed into a predictive model! The final step is to choose which predictive model to use. There are hundreds to choose from, and trade-offs associated with each. The good news is, as we alluded to earlier there are different types of machine learning problems and each come with their own set of algorithms – so this narrows down our search considerably.

In our case, we're working on a supervised classification problem, so the upper left hand corner displays some common algorithms for problems like ours. It's very common to test a handful of them and choose the one that performs best on your dataset (or even combine some of them into an ensemble model.)

Remember, the ultimate objective to choose a model that learns a predictive rule (which comes in the form of an equation) that can be generalized to new observations (both for classification and regression)

Bias-Variance Tradeoff



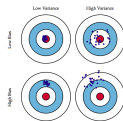
Machine Learning From Scratch

Steps in the Machine Learning Process

Step 4: Model Selection

Bias-Variance Tradeoff

Bias-Variance Tradeoff



One critical aspect to take into consideration when choosing a model is called the bias-variance tradeoff

So first, I'm going to define what I mean when I talk about the terms "bias" and "variance" in the context of machine learning, then we are going to look at some simplified examples (using this dataset here) in the regression context that I think really well convey the essence of applied machine learning:

Maybe still show my graphs?

mention statistical techniques necessary to combat variance, like bagging

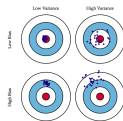
maybe mention regularization?

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Bias-Variance Tradeoff

Bias-Variance Tradeoff



Throughout this section, add notes for each of the headers in the jupyter notebook!

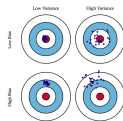
Also, do not forget to look over Lin alg for SVM notebook!efz

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Bias-Variance Tradeoff

Bias-Variance Tradeoff



Here we'll first be going through a general example of building an svm from scratch on a toy dataset, then apply some of Python's ml tools to our crash data set to predict whether or not a car crash was caused by a drunk driver!

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Here, we will begin by importing the necessary libraries – (audience should just run these). So here is what our plotted data looks like. And here is a snapshot of data set. Note that for each data point (each row/each dot), we have an x_1 value, and X_2 value, and a y value (which represents the target variable, which we are trying to predict). In this case, we've rather defined the red data points to be labeled as -1 and the blue data points to be labeled as 1. It may seem weird to call them x_1 and x_2 , but we'll address why we do that in a little bit.

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Representation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Representation: Support Vector Machine

The first step in building any sort of predictive model is choosing a representation. For our example today, we choose a Support Vector Machine. Big picture - The goal of a SVM is to To find the optimal separating hyperplane which maximizes the margin of the training data.

Now, a hyperplane is kind of like a fancy word for a line, or maybe better put, it's a general way to talk about a line. The black line on this graph of our dataset is a hyperplane that separate the two classes (red/blue). Since our data is two dimensional (we have an x , and a y (1, 2)) the hyperplane is a line

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Representation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

How do we find the optimal separating hyperplane?:

Now, as we mentioned, the goal of the hyperplane is to find the optimal separating hyperplane, so how do we do that? It appears that there can be many different separating hyperplanes (often there is infinite). We're going to choose the hyperplane that is as far away as possible from each class of datapoints.

We want to maximize the margin because It generalizes better to classifying unseen observations - \hat{y} (meaning that it makes better predictions)

If we have our usual equation for a line: $y=mx+b$, then all we need to do is find that optimal m and b that characterize the optimal hyperplane!

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Representation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Understanding the Definition of a hyperplane:

As we've seen previously, the separating hyperplane that we keep referring to (in 2-d) is just a line. I'm confident that you're familiar with the way we define lines:

Now, a quick refresher that b is just a constant that represents the y-intercept, m (usually defines the slope) is a coefficient to the variable x (also a constant) and the variable y even has a coefficient as well, in this case it's 1. if we try to expand this equation $y=mx+b$ to more and more dimensions(variables), it's not a real great form to express it in. So instead, we choose to set the whole equation equal to zero. With some algebraic manipulation, we get that:

So, that looks a bit unnatural..so its common convention to represent all coefficients/constants with β , and all variables with X_i , so that we don't run out of variables to use.

In our Crash data, we use 14 variables...

maybe change the betas to w 's?

2018-09-05

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GRU Crash dataset

Evaluation

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Quick refresher on the dot product:

Now, we're going to talk for a minute about the dot product. The dot product has many different names across fields of mathematics: inner product, and linear combination are also common. Regardless of its name, it is a useful operation to use between vectors. To put it simply, if you take the dot product of two vectors, you multiply all their corresponding elements and take the sum. (note that the vectors must have the same dimensions).

The output of the dot product is not another vector, but just a scalar value.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Dot product as distance from hyperplane:

Now you'll notice that this dot product indeed looks very similar to the way that we've defined the equation of a hyperplane. It turns out that, when we take the dot product of our weights vector \vec{w} and a single datapoint $\vec{x}^{(i)}$, the resulting number that we get can be thought of as the distance from the data point to the hyperplane (how far away it is). And since we're trying to maximize the width of our margin, having an efficient way to calculate how far away each datapoint is from the hyperplane proves to be crucial.

Now, if we're being mathematically rigorous, we may want to frame that a bit differently, but at least for all intensive purposes, we can think of it this way!

The fact that we've defined the equation of a hyperplane to be the dot product of our weights and features equal to zero implies that the hyperplane and the weights vectors are perpendicular (aka orthogonal).

And since they are orthogonal, we're able to exploit some linear algebra

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

The Hinge Loss Function is defined as:

Now, armed with this ability to calculate how far away each data point is from the hyperplane, we can now use the Hinge Loss Function to quantify how wrong each of our predictions are!

One more important thing to note about the dot product is that it is signed, meaning that it can be either positive or negative, namely, data points below the hyperplane will have a negative distance from the hyperplane, and datapoints above the hyperplane, will have a positive distance from hyperplane.

Talk about how we will predict which class each data point comes from, then use the hinge loss to tell us how wrong we are.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

The Hinge Loss Function is defined as (Part 2): Breaking it down
1 minus y times the dot product of the weights vector and the datapoint itself.

Let's break this down piece by piece. First, we'll look at the inputs. \vec{w} is a vector of weights (coefficients) in the linear equation. $\vec{x}^{(i)}$ is a vector representing a single datapoint (the i^{th} datapoint), or a single row in our dataset. $y^{(i)}$ is the label associated with the datapoint (which is either 1 or -1). The output is a scalar value (a number) representing a penalty for how wrong our prediction was. The greater the penalty, the worse our estimated weights were in classifying the data.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to G81 Crash dataset

The Hinge Loss Function is defined as (Part 3): little subscript plus sign

One thing I've neglected to mention thus far is the little subscript plus sign at the end of the loss function. That little plus sign denotes that our loss function only looks at positive values, meaning that the output of function is a negative value, it's just going to change it to zero. This intuitively makes sense! If we get a negative penalty for our algorithm... that means that the prediction must be very correct, so instead, we'll just assign a penalty of 0, basically no penalty at all.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

The Hinge Loss Function is defined as (Part 4): working it out

Now, if you actually try to work this out with the datapoints, it can be a bit confusing, as there are lots of negatives and the sign of the function is constantly changing, so I've included this little table that works out small examples for all 4 cases, when the data point is correctly and incorrectly classified from the negative class, and when the data point was correctly and incorrectly classified from the positive class. For these little examples, I've assumed that the datapoint is 3 units away from the hyperplane. You'll notice if you skip to the bottom that the penalty for both incorrectly classified datapoints is 4, while the penalty for the correctly classified datapoints is 0.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

The Hinge Loss Function is defined as (Part 5): essence of ml

This is really worth emphasizing, because it doesn't just apply to SVM's but is the basis of machine learning. Machines learn by associating a quantitative penalty to incorrect predictions. The machine then sets out to minimize the penalty, which ultimately will result in making the maximum amount of correct predictions.

This is the essence of the current state of artificial intelligence. All AI is based on this principle, everything from autonomous vehicles, to amazon's recommendation system.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Evaluation

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Hinge Loss with regularization term:

Now, with the hinge loss function we've defined, when we try to minimize the loss function, we will (if possible) find a separating hyperplane that perfectly classifies the data i.e. no misclassifications. Which sounds great! The only problem with that is that if there are any outliers in our dataset, it can seriously skew the hyperplane. Below we have two nearly identical datasets...the one on the right has one outlier (circle it with mouse) and it has completely changed our hyperplane.

Again, we need to remember that our goal is to find a hyperplane that will best classify new datapoints...so we don't want it to be affected by outliers.

The λ coefficient acts as a tuning parameter (in our example, we will explicitly hard-code a value for lambda), but generally, as the value for lambda gets smaller and smaller (approaches 0), the margin grows wider (to the point of misclassifying data points). As the value for lambda gets larger and larger, the margin will get smaller and smaller (to the point of

2018-09-05

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GRU Crash dataset

Optimization

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Optimization

Now, we've been talking quite a bit about minimizing the penalty to our algorithm for mis-classifying data points, but it is not immediately obvious how to minimize the penalty.

We're going to use an algorithm called Stochastic Gradient Descent that will iteratively find the weights that minimize the total "loss" or "cost" to our Support Vector Machines, resulting in the very best possible predictions. **Look at this visually in two ways! 1. With stanford web demo, 2. with gradient descent drawing**

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Stanford Web App

For one of its course, Stanford has created this awesome web demo that actively animates gradient descent. Here, they have three different classes (red, blue, green), while we only have two (red, blue), yet the principles are exactly the same. Gradient Descent begins with totally random separating boundaries, this means that we begin with random weights, \vec{w} , random coefficients in the equation of the separating hyperplane. (click randomize a few times)

The algorithm then iteratively updates the weights little by little until the separating hyperplane is correctly classifying the data points.(start repeated update with softmax)

So this is how gradient descent works.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

- Step 0: Identify The Problem
- Step 1: Get the Data
- Step 2: Data Exploration
- Step 3: Data Preparation
- Step 4: Model Selection
- Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

- Representation
- Evaluation
- Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Gradient Descent Drawing

I want to show you another angle we can look at this from. Take a look at this drawing. Imagine that this red line is the hinge loss function (preface: it's not. But it is a representative of the general concept of gradient descent), which, remember, outputs how wrong our predictions are, so we want to minimize this function, by finding the weights that correspond to the smallest possible output. All we have to do is move towards the bottom of this function, which is exactly what gradient descent does.

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to Gilt Crash dataset

Transition to Math of GD

Pragmatically speaking, the algorithm is going to loop through each observation (row) in our dataset, and check if our current weights would have correctly or incorrectly classified it. If it would have incorrectly classified the data point, then the algorithm updates the weights accordingly.

A natural question to ask then is, how are we going to update the weights? How do we move from the top of the function to the bottom? We will do so by using partial derivatives of the hinge loss function. We do this because:

- The derivative of the hinge loss function gives us the slope of our hinge loss function at our current values for \vec{w} . Given the slope, we want to "move" or update our weights in the direction that the slope is decreasing i.e heading towards the bottom of the function.

Machine Learning From Scratch

- └ Building a Support Vector Machine from Scratch
 - └ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GIt Crash dataset

Calculating the partial derivatives:

Okay, so let's calculate the partial derivatives. Using a rule from calculus, we can separately calculate the partial derivative of the first and second terms of the hinge loss function (loss term and regularization term) separately.

As we can see here, the partial derivative of the loss term depends on whether the datapoint was correctly or incorrectly classified. (0 if correctly classified, or $-y$ times x if incorrectly classified). The partial derivative of the regularization term is 2 times λ times w .

2018-09-05

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GRU Crash dataset

Update Rules:

2018-09-05

Machine Learning From Scratch

└ Building a Support Vector Machine from Scratch

└ Optimization

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to G81 Crash dataset

To do next:

Watch Andrew Ng's explanation of stochastic gradient descent and take notes in order to explain!

Then also outline exactly how to calculate the gradient of the hinge loss function for misclassified and correctly classified datapoints, then explicitly show the update rules.

Machine Learning Overview

Steps in the Machine Learning Process

Step 0: Identify The Problem

Step 1: Get the Data

Step 2: Data Exploration

Step 3: Data Preparation

Step 4: Model Selection

Step 5: Cross-validation/Hyper-parameter tuning

Building a Support Vector Machine from Scratch

Representation

Evaluation

Optimization

Exploring Scikit-Learn and applying to GR Crash dataset



<https://www.sisense.com/glossary/data-exploration/>



<https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63>



<http://scott.fortmann-roe.com/docs/BiasVariance.html>