Universidad Zaragoza

Machine Learning (69152)

Machine Learning Fundamentals

Universidad Zaragoza

Javier Civera
jcivera@unizar.es, office 3.01, Ada Byron building
Dpto. Informática e Ingeniería de Sistemas.

Happy to be here!

Content

DISCLAIMER: SLIDES WILL BE UPDATED DURING THE COURSE, DOWNLOAD THE MOST UPDATED VERSION FROM MOODLE

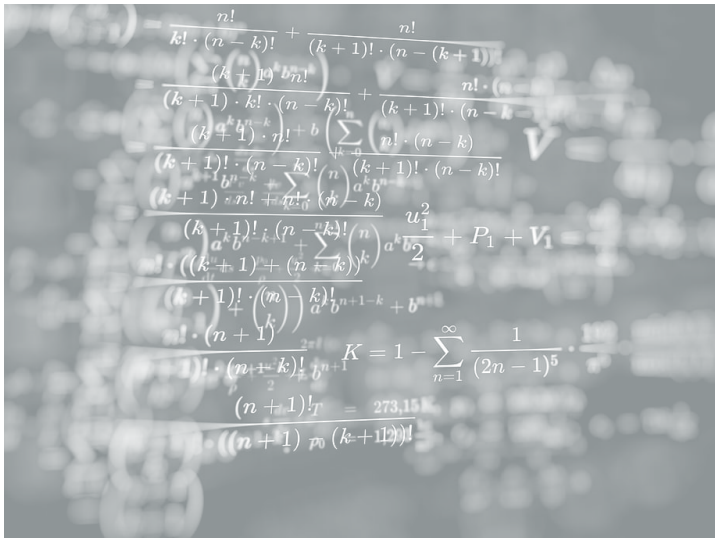
- **Introduction**
- **K-nearest neighbours (as a simple example)**
- **Regression**
    - Linear regression
    - Non-linear regression
    - Overfitting, regularization, model selection
    - Robust regression
    - Ridge regression
- **Classification**
    - Logistic regression
    - SVM
- **Unsupervised methods**
    - PCA
- **Evaluation metrics**

Bibliography

- **Kevin P. Murphy, Probabilistic Machine Learning: An Introduction, The MIT Press, 2022** (available online https://probml.github.io/pml-book/book1.html)
- Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong, Mathematics for Machine Learning, Cambridge University Press, 2020 (available online https://mml-book.com/)
- Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006 (available online https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf)
- Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, O'Reilly, 2019 (2nd edition)
- Online courses
  - Andrew Ng, Stanford CS229: Machine Learning, https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU
  - Ayush Singh and @freecodecamp, ML for beginners, https://www.youtube.com/watch?v=NWONeJKn6kc
- Python resources
  - The Python tutorial: https://docs.python.org/3/tutorial/
  - numpy tutorial: https://cs231n.github.io/python-numpy-tutorial/
  - scikit-learn tutorials: https://scikit-learn.org/stable/tutorial/index.html

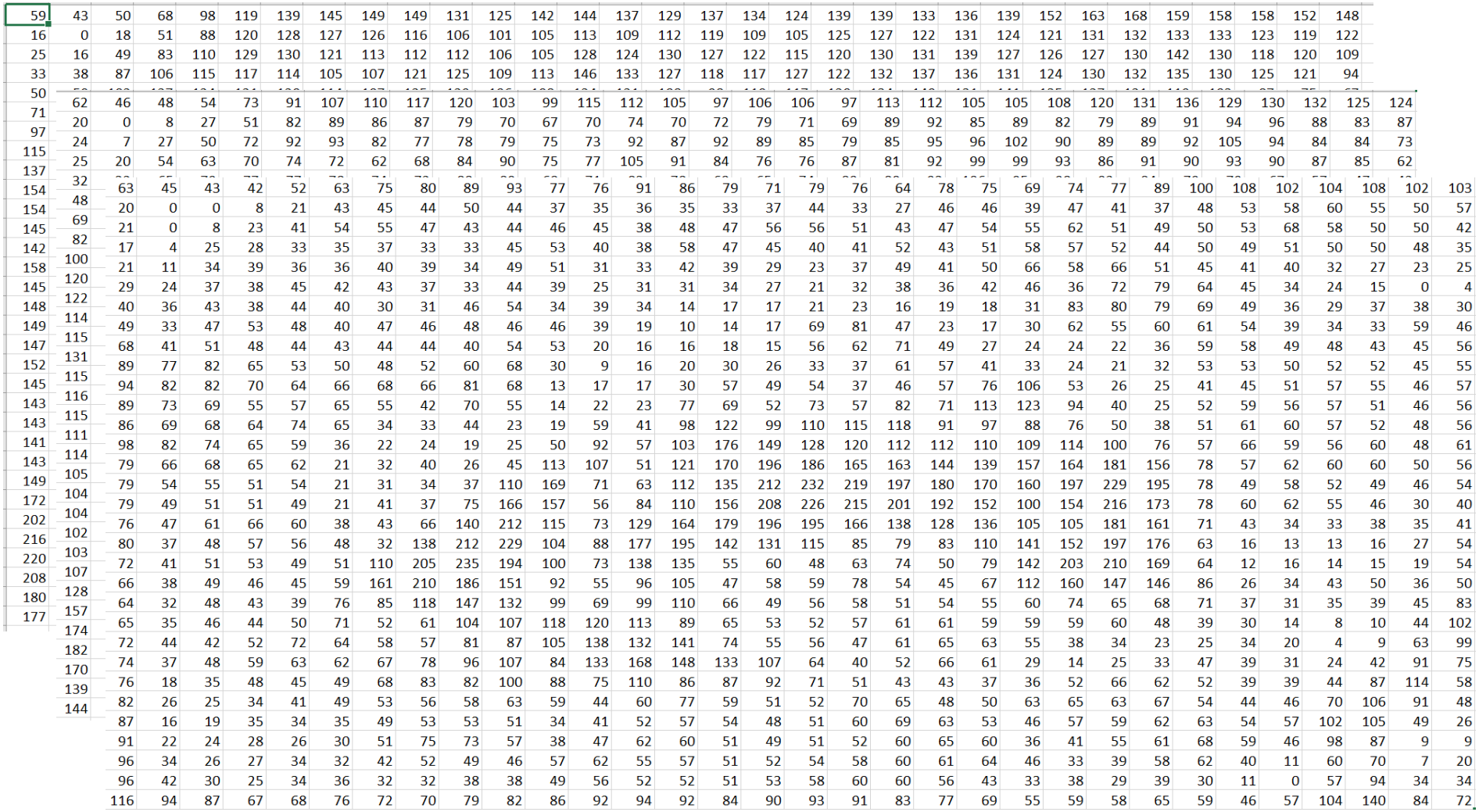
Motivation

- Machine learning has a huge presence (and growing)…
  - Recommenders, virtual assistants, content generation (intelligent virtual characters, automatic scene creation), spam filtering, search engines, social media, news classification, video surveillance, face recognition, fraud detection, traffic prediction, image retrieval, predictive maintenance, weather forecasting, autonomous driving, speech recognition, automatic translation, medical diagnosis…
- Starting by the fundamentals (and getting deep into math!) is critical for a structured understanding of the field.



Harder than it seems! → a CV example

What is this?

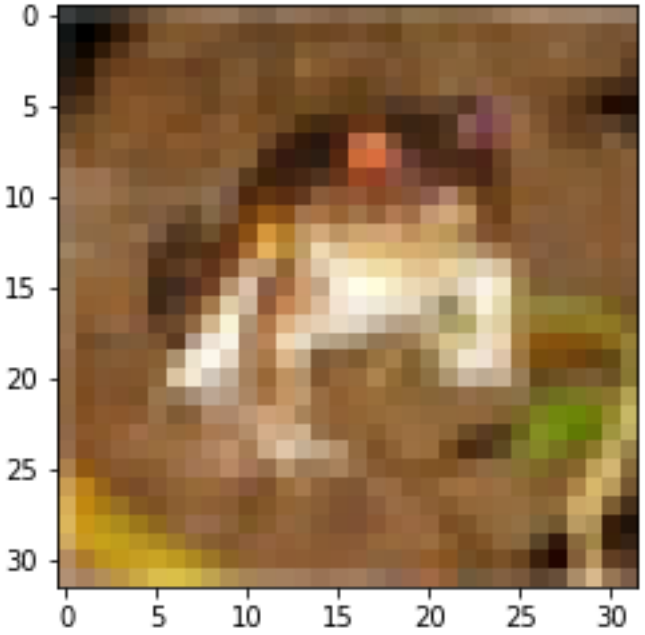


Harder than it seems! → a CV example

What is this? → humans excel at recognizing certain patterns that are hard for machines…

59	43	50	68	98	119	139	145	149	149	131	125	142	144	137	129	137	134	124	139	139	133	136	139	152	163	168	159	158	158	152	148
16	0	18	51	88	120	128	127	126	116	106	101	105	113	109	112	119	109	105	125	127	122	131	124	121	131	132	133	133	123	119	122
25	16	49	83	110	129	130	121	113	112	112	106	105	128	124	130	127	122	115	120	130	131	139	127	126	127	130	142	130	118	120	109
33	38	87	106	115	117	114	105	107	121	125	109	113	146	133	127	118	117	127	122	132	137	136	131	124	130	132	135	130	125	121	94
50	59	102	127	124	121	120	114	107	125	129	106	108	124	121	108	98	110	117	120	134	140	131	141	135	127	121	119	103	87	75	67
71	84	110	129	136	131	129	119	108	122	123	105	107	111	108	98	94	97	83	88	102	97	88	118	140	136	120	107	88	67	35	32
97	111	123	130	136	132	122	121	127	138	124	120	107	80	68	74	101	105	65	58	63	78	136	122	139	151	129	108	95	96	89	66

115	119	62	46	48	54	73	91	107	110	117	120	103	99	115	112	105	97	106	106	97	113	112	105	105	108	120	131	136	129	130	132	125	124
137	128	20	0	8	27	51	82	89	86	87	79	70	67	70	74	70	72	79	71	69	89	92	85	89	82	79	89	91	94	96	88	83	87
154	154	24	7	27	50	72	92	93	82	77	78	79	75	73	92	87	92	89	85	79	85	95	96	102	90	89	89	92	105	94	84	84	73
154	155	25	20	54	63	70	74	72	62	68	84	90	75	77	105	91	84	76	76	87	81	92	99	99	93	86	91	90	93	90	87	85	62
145	146	32	32	65	79	77	77	78	74	72	88	89	68	71	83	78	68	65	74	80	80	93	106	95	98	92	84	79	79	67	57	47	42
142	141	48	53	73	82	88	84	84	77	70	82	81	65	72	77	74	65	62	63	56	58	68	69	54	74	96	97	80	68	54	39	10	13
158	154	69	75	85	84	88	83	74	74	83	94	79	79	71	50	43	41	51	56	37	36	37	51	93	68	86	106	87	68	59	63	61	47
145	149	82	76	63	45	43	42	52	63	75	80	89	93	77	76	91	86	79	71	79	76	64	78	75	69	74	77	89	100	108	102	104	108	102	103
148	146	100	82	20	0	0	8	21	43	45	44	50	44	37	35	36	35	33	37	44	33	27	46	46	39	47	41	37	48	53	58	60	55	50	57
149	143	120	112	21	0	8	23	41	54	55	47	43	44	46	45	38	48	47	56	56	51	43	47	54	55	62	51	49	50	53	68	58	50	50	42
147	134	122	117	17	4	25	28	33	35	37	33	33	45	53	40	38	58	47	45	40	41	52	43	51	58	57	52	44	50	49	51	50	50	48	35
152	117	114	109	21	11	34	39	36	36	40	39	34	49	51	31	33	42	39	29	23	37	49	41	50	66	58	66	51	45	41	40	32	27	23	25
145	127	115	106	29	24	37	38	45	42	43	37	33	44	39	25	31	31	34	27	21	32	38	36	42	46	36	72	79	64	45	34	24	15	0	4
143	127	131	119	40	36	43	38	44	40	30	31	46	54	34	39	34	14	17	17	21	23	16	19	18	31	83	80	79	69	49	36	29	37	38	30
143	125	115	109	49	33	47	53	48	40	47	46	48	46	46	39	19	10	14	17	69	81	47	23	17	30	62	55	60	61	54	39	34	33	59	46
141	131	116	100	68	41	51	48	44	43	44	44	40	54	53	20	16	16	18	15	56	62	71	49	27	24	24	22	36	59	58	49	48	43	45	56
143	139	115	95	89	77	82	65	53	50	48	52	60	68	30	9	16	20	30	26	33	37	61	57	41	33	24	21	32	53	53	50	52	52	45	55
149	133	111	88	94	82	82	70	64	66	68	66	81	68	13	17	17	30	57	49	54	37	46	57	76	106	53	26	25	41	45	51	57	55	46	57
172	144	114	75	89	73	69	55	57	65	55	42	70	55	14	22	23	77	69	52	73	57	82	71	113	123	94	40	25	52	59	56	57	51	46	56
202	187	105	82	86	69	68	64	74	65	34	33	44	23	19	59	41	98	122	99	110	115	118	91	97	88	76	50	38	51	61	60	57	52	48	56
216	193	104	80	98	82	74	65	59	36	22	24	19	25	50	92	57	103	176	149	128	120	112	112	110	109	114	100	76	57	66	59	56	60	48	61
220	201	104	76	79	66	68	65	62	21	32	40	26	45	113	107	51	121	170	196	186	165	163	144	139	157	164	181	156	78	57	62	60	60	50	56
208	201	102	80	79	54	55	51	54	21	31	34	37	110	169	71	63	112	135	212	232	219	197	180	170	160	197	229	195	78	49	58	52	49	46	54
180	173	103	87	79	49	51	51	49	21	41	37	75	166	157	56	84	110	156	208	226	215	201	192	152	100	154	216	173	78	60	62	55	46	30	40
177	168	107	80	76	47	61	66	60	38	43	66	140	212	115	73	129	164	179	196	195	166	138	128	136	105	105	181	161	71	43	34	33	38	35	41
		128	88	80	37	48	57	56	48	32	138	212	229	104	88	177	195	142	131	115	85	79	83	110	141	152	197	176	63	16	13	13	16	27	54
		157	129	72	41	51	53	49	51	110	205	235	194	100	73	138	135	55	60	48	63	74	50	79	142	203	210	169	64	12	16	14	15	19	54
		174	136	66	38	49	46	45	59	161	210	186	151	92	55	96	105	47	58	59	78	54	45	67	112	160	147	146	86	26	34	43	50	36	50
		182	150	64	32	48	43	39	76	85	118	147	132	99	69	99	110	66	49	56	58	51	54	55	60	74	65	68	71	37	31	35	39	45	83
		170	153	65	35	46	44	50	71	52	61	104	107	118	120	113	89	65	53	52	57	61	61	59	59	59	60	48	39	30	14	8	10	44	102
		139	123	72	44	42	52	72	64	58	57	81	87	105	138	132	141	74	55	56	47	61	65	63	55	38	34	23	25	34	20	4	9	63	99
		144	129	74	37	48	59	63	62	67	78	96	107	84	133	168	148	133	107	64	40	52	66	61	29	14	25	33	47	39	31	24	42	91	75
				76	18	35	48	45	49	68	83	82	100	88	75	110	86	87	92	71	51	43	43	37	36	52	66	62	52	39	39	44	87	114	58
				82	26	25	34	41	49	53	56	58	63	59	44	60	77	59	51	52	70	65	48	50	63	65	63	67	54	44	46	70	106	91	48
				87	16	19	35	34	35	49	53	53	51	34	41	52	57	54	48	51	60	69	63	53	46	57	59	62	63	54	57	102	105	49	26
				91	22	24	28	26	30	51	75	73	57	38	47	62	60	51	49	51	52	60	65	60	36	41	55	61	68	59	46	98	87	9	9
				96	34	26	27	34	32	42	52	49	46	57	62	55	57	51	52	54	58	60	61	64	46	33	39	58	62	40	11	60	70	7	20
				96	42	30	25	34	36	32	32	38	38	49	56	52	52	51	53	58	60	60	56	43	33	38	29	39	30	11	0	57	94	34	34
				116	94	87	67	68	76	72	70	79	82	86	92	94	92	84	90	93	91	83	77	69	55	59	58	65	59	46	57	104	140	84	72

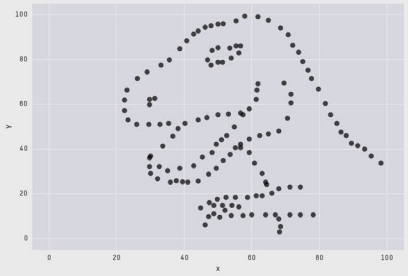


Mach bands,
Ernst Mach, 1865

Universidad Zaragoza
1542

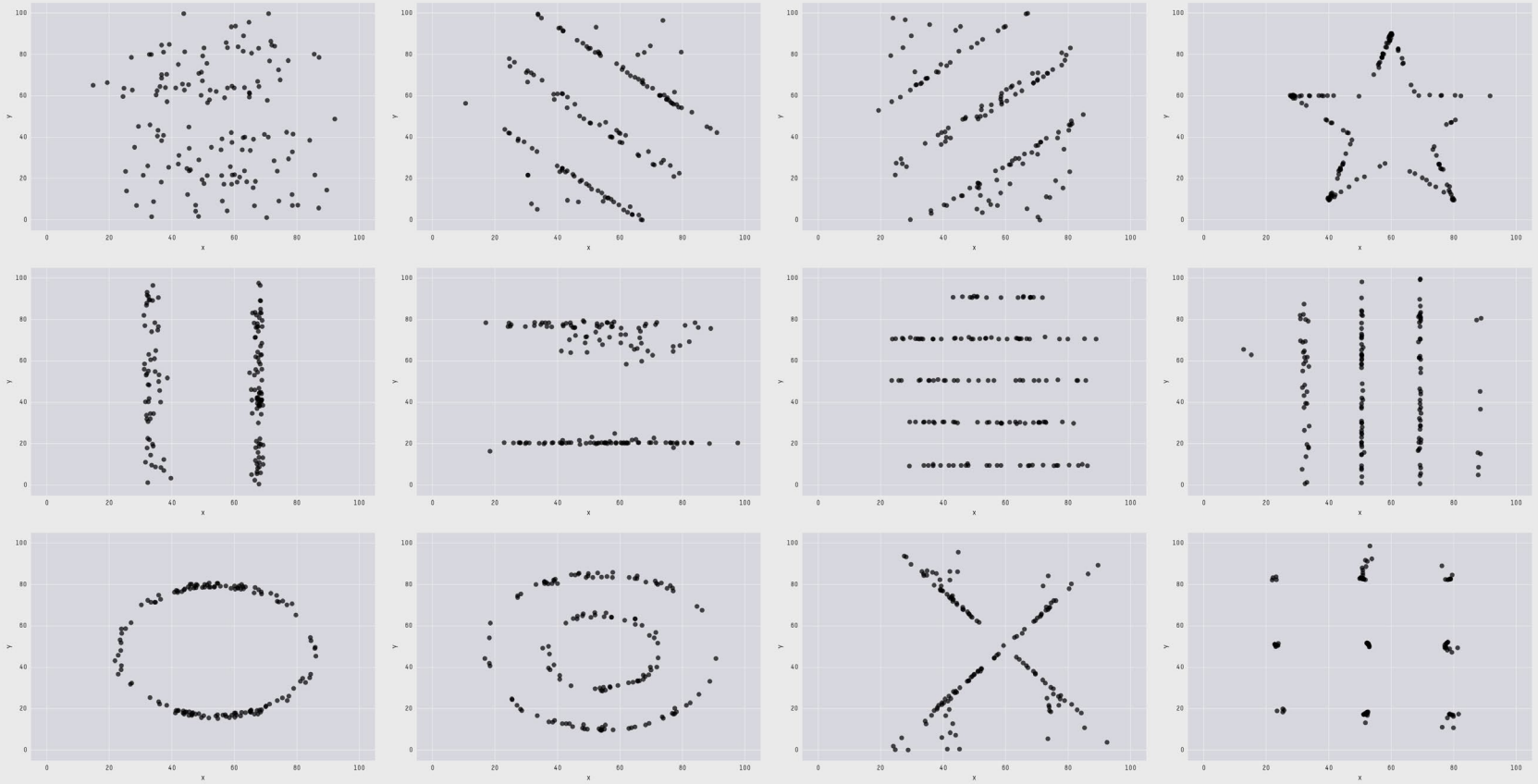
Harder than it seems!

Characterizing our data: what do we model? What is noise?



X Mean: 54.26
Y Mean: 47.83
X SD : 16.76
Y SD : 26.93
Corr. : -0.06

Alberto Cairo, the Datasaurus dataset



Machine learning is NOT magic!

- Careful methodology, iterative design and data analysis can be motivated by the No Free Lunch theorem [Wolpert 96].
  - A model is a simplified version of the data
  - Assumptions (~hyperparameters) are needed (what should be accounted and what should be ignored in the data)
  - No model is guaranteed a priori to be better than others, it just should "fit" our data
  - Usual practice: Evaluate a few reasonable models making reasonable assumptions on your data.

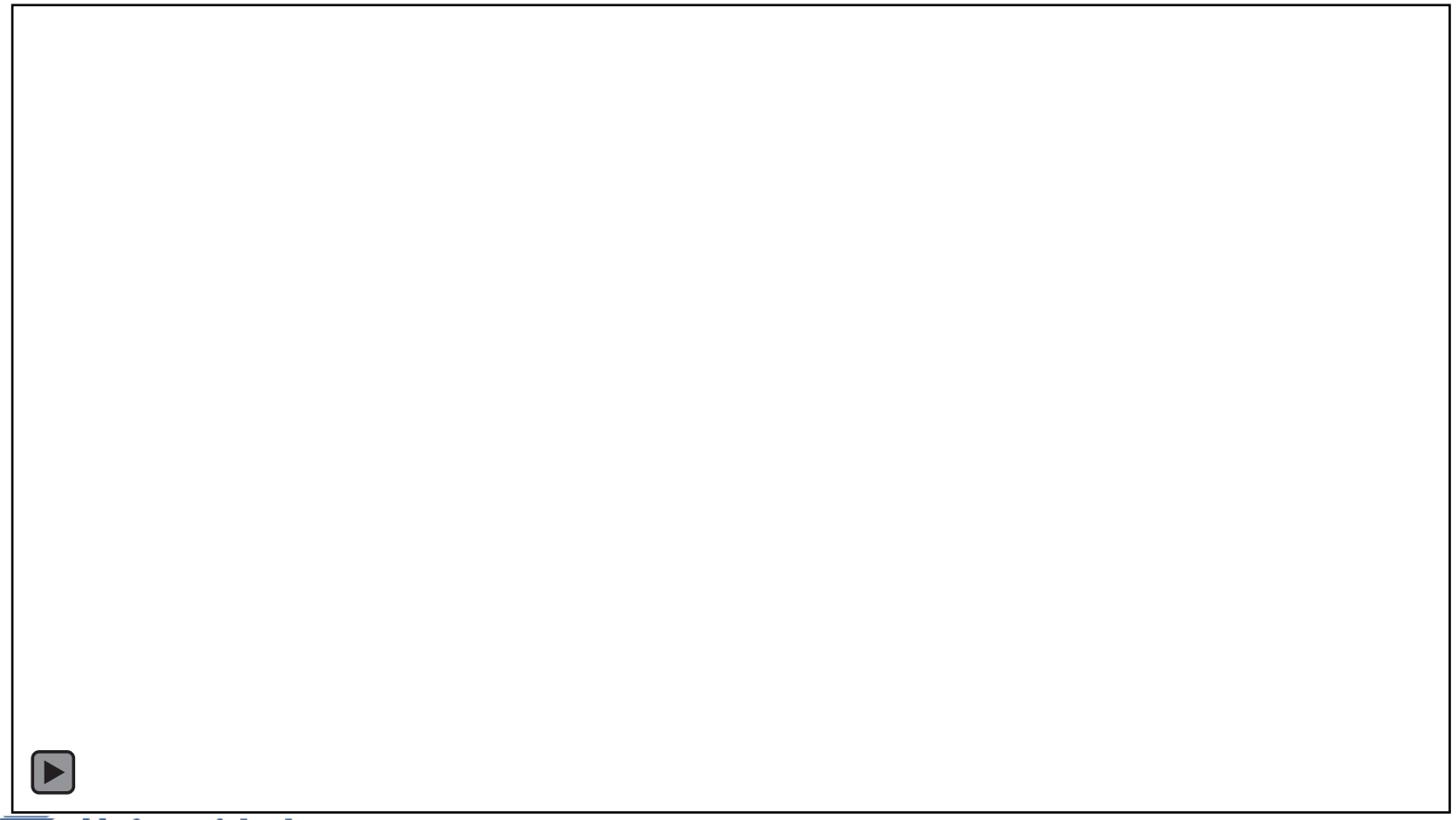
Sometimes it seems magical though…

- **Example:** Neural Radiance Fields
  - Use neural networks (MLPs) for representing scenes and rendering novel views, costly to train and query

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, NeRF: Representing scenes as neural radiance fields for view synthesis, ECCV 2020
Thomas Müller and Alex Evans and Christoph Schied and Alexander Keller, Instant Neural Graphics Primitives with a Multiresolution Hash Encoding, ACM ToG, 2022

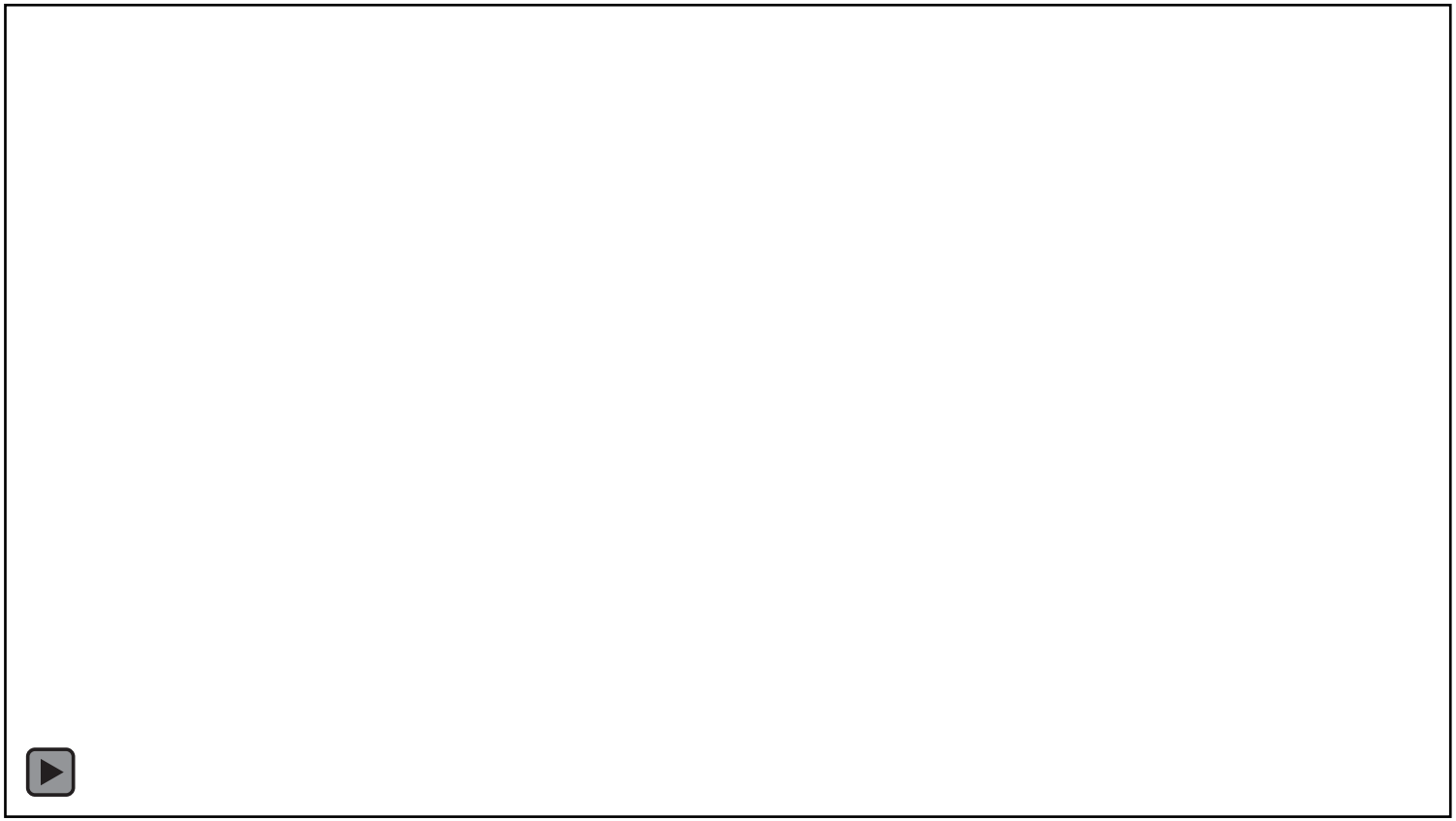
Sometimes it seems magical though…

- **Example:** Neural Radiance Fields
  - Use neural networks (MLPs) for representing scenes and rendering novel views, costly to train and query



Shallow learning also works!

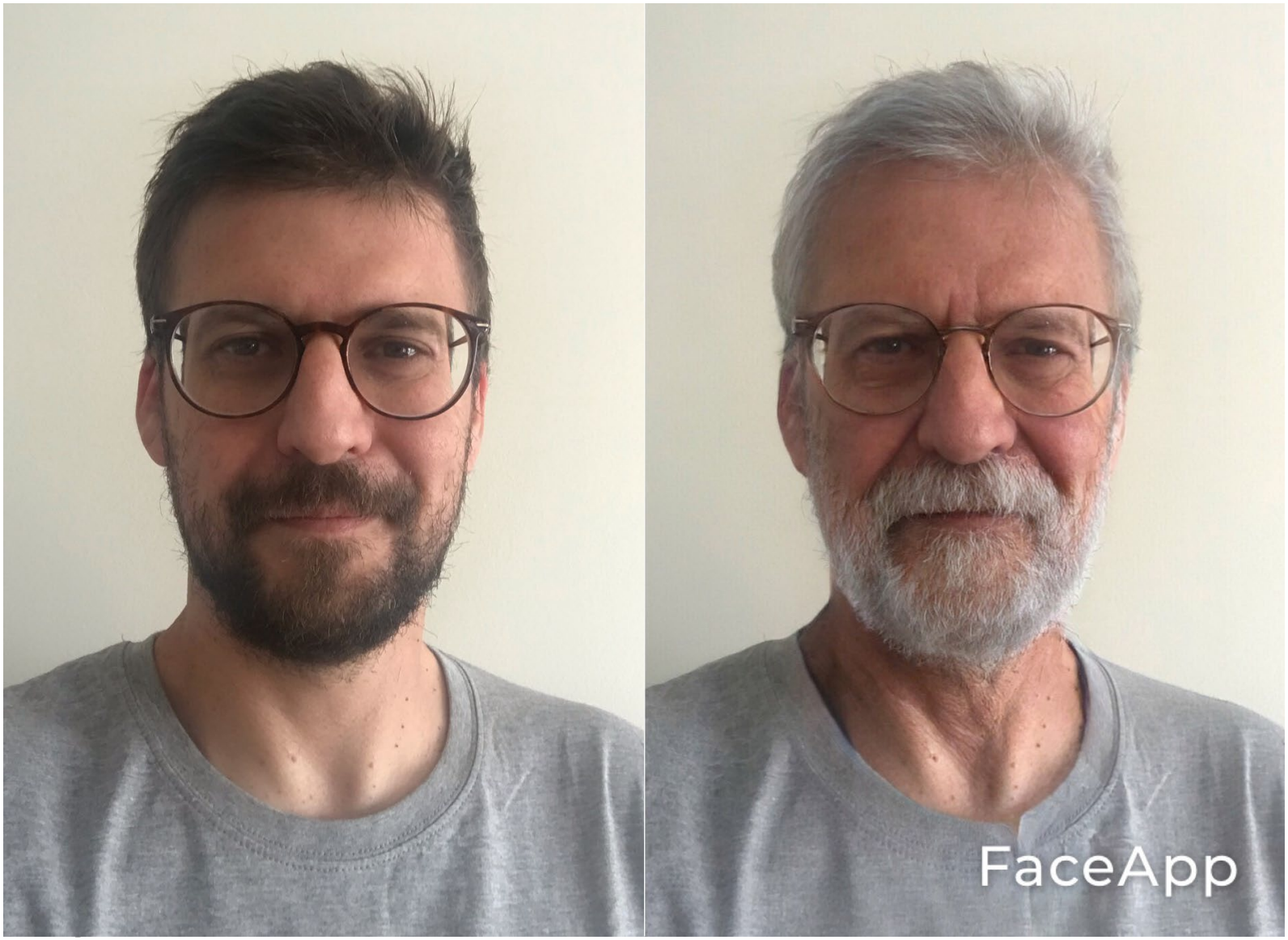
- Scene representation based on Gaussians in the 3D space
- Well implemented but standard point-based rendering
- Enables real-time capabilities at 1080p



Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G., 3D Gaussian Splatting for real-time radiance field rendering, ACM ToG, 2023

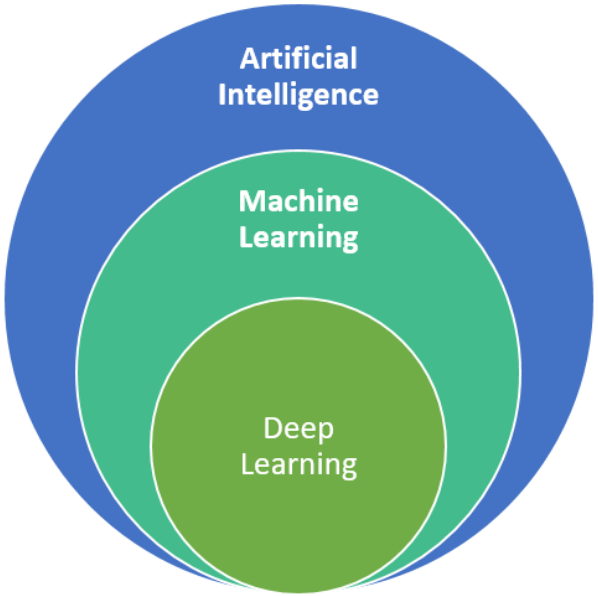
Sometimes it seems magical though…

Just one cool example of sooo many…



Definitions

- **Machine Learning:**
  - Field of study that gives computers the ability to learn without being explicitly programmed (Arthur Samuel 1959).
  - A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. (Tom Mitchell, 1997)
  - Success stories: Spam filtering, OCR, face detection, search engines…
- **Related terms:**
  - **Data mining:** The field of study that aims to extract information and structure from raw data.
  - **Artificial Intelligence:** the theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.
  - **Knowledge Discovery**, **Pattern Recognition**, …



Taxonomy

- **Training set:** Data samples that machine learning algorithms use to find the model parameters.
- **Test set:** Data used to assess generalization over samples that are unseen during training.
- **Supervised Learning:** The training set is composed of **labeled data** (data + desired output)
- **Unsupervised Learning:** The training data is **unlabeled**.
- **Regression:** Estimates a function that relates an input vector with a **continuous output**.
- **Classification**: Estimates a function that relates an input vector with a **discrete output**.
- **Parametric:** The model has a fixed number of parameters (by making assumptions on the data distribution)
- **Non-parametric:** The number of parameters grow with the data

Methodology (I de III)

1. Frame the Problem (Look at the Big Picture)

- Define the general high-level objective
- Frame the problem (supervised/unsupervised, regression/classification, etc.)
- Define metrics to evaluate performance (e.g., error, accuracy, false positives rate, etc.)
- What is a minimum/reasonable/maximum performance? Check scientific literature and existing similar systems.

2. Get the data

- What data is needed? How much? Which format?
- How/where do you get the data? Check legal aspects and sensitive information!
- Divide into training/validation/test sets. Never look at the test set!

Methodology (II de III)

3. Explore the data

- Visualize the data
- Study correlations between atributes

4. Prepare the data

- Work in a copy of the data, keep the original data intact
- Data cleaning: Fix/remove outliers, fill missing values / remove rows
- Feature selection: Drop uninformative attributes (garbage in – garbage out)
- Feature engineering: Transform the data into more descriptive/separable features (e.g., #murders vs. #murders/population)
- Feature scaling: Standardize or normalize features/data
- Shuffle the data

Methodology (III de III)

5. Shortlist promising models

- Sample subsets of the training sets for quick exploration of many techniques/models
- Compare models using cross-validation (mean and std over folds)
- Analyze errors

6. Fine-tune the system

- Fine-tune hyperparameters using cross-validation
- Try ensemble methods
- Once finished, evaluate the performance on the test set

7. Present the solution

- Report relevant aspects of the process, not only performance
- Failure cases might be relevant!
- Report assumptions, limitations, how the model could be improved
- Summarize (graphs, key findings)

K-nearest-neighbor (birds of a feather…)

- We have N sample pairs in our training dataset $\mathcal{D}$

$$\mathcal{D} = \left\{\left(x^{(1)}, y^{(1)}\right), \cdots, \left(x^{(i)}, y^{(i)}\right), \cdots, \left(x^{(N)}, y^{(N)}\right)\right\}$$

- 1-NN: for a query sample $x^{(q)}$, we predict the $y^{(q)}$of the closest element in our dataset $\mathcal{D}$

$$y^{(q)} = y^{(p)}, \qquad p = \arg min_i(d(x^{(q)}, x^{(i)}))$$

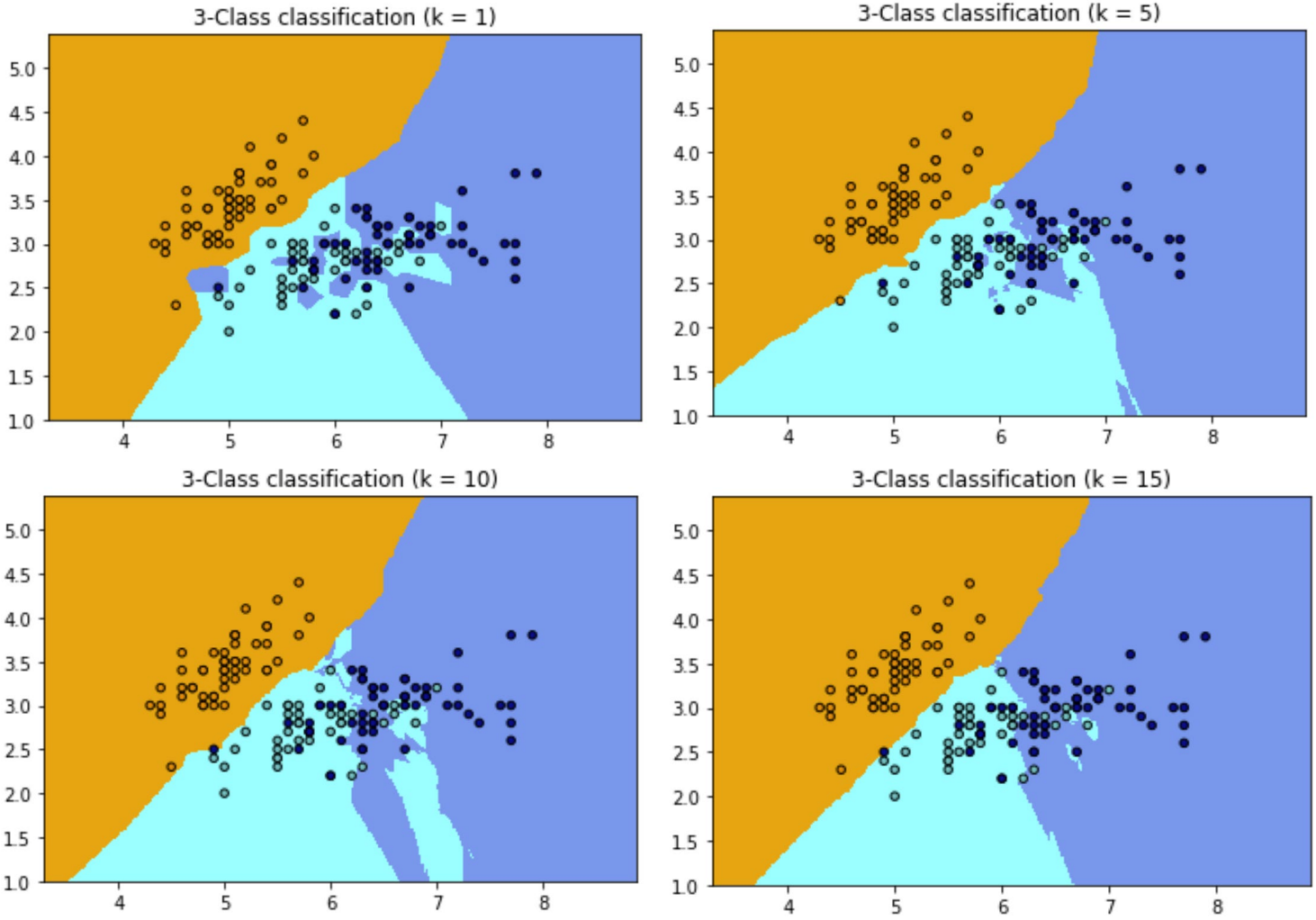
  - $d(\cdot)$ distance metric (usually Euclidean, can be learnt)
  - Supervised, classification/regression, non-parametric ("lazy", keeps the data instead of summarizing it into a model)
  - In k-NN (usually k even), the k-nearest neighbours vote for the class
  - Very simple and intuitive.
  - It works reasonably well if the feature space is reasonably well sampled by $\mathcal{D}$
  - Computationally demanding at test time!

Let's start coding!

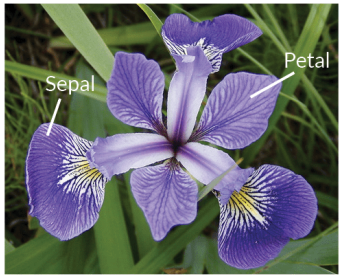
- If you are not familiar with Python, **do the tutorial** https://docs.python.org/3/tutorial/
- To make things easier at the beginning, use Google Colab https://colab.research.google.com/
- Use sklearn documentation https://scikit-learn.org/ and google for help
- The ML community shares a lot of resources, enjoy!

K-nn examples – Iris

- Iris, sepal length and width, k={1, 5, 10, 15}



Iris Setosa



Iris Versicolor

Iris Virginica

Colab link:
https://colab.research.google.com/drive/1Dyov9NO9ySNi3vSphk1JZJzlyqJoF3EG

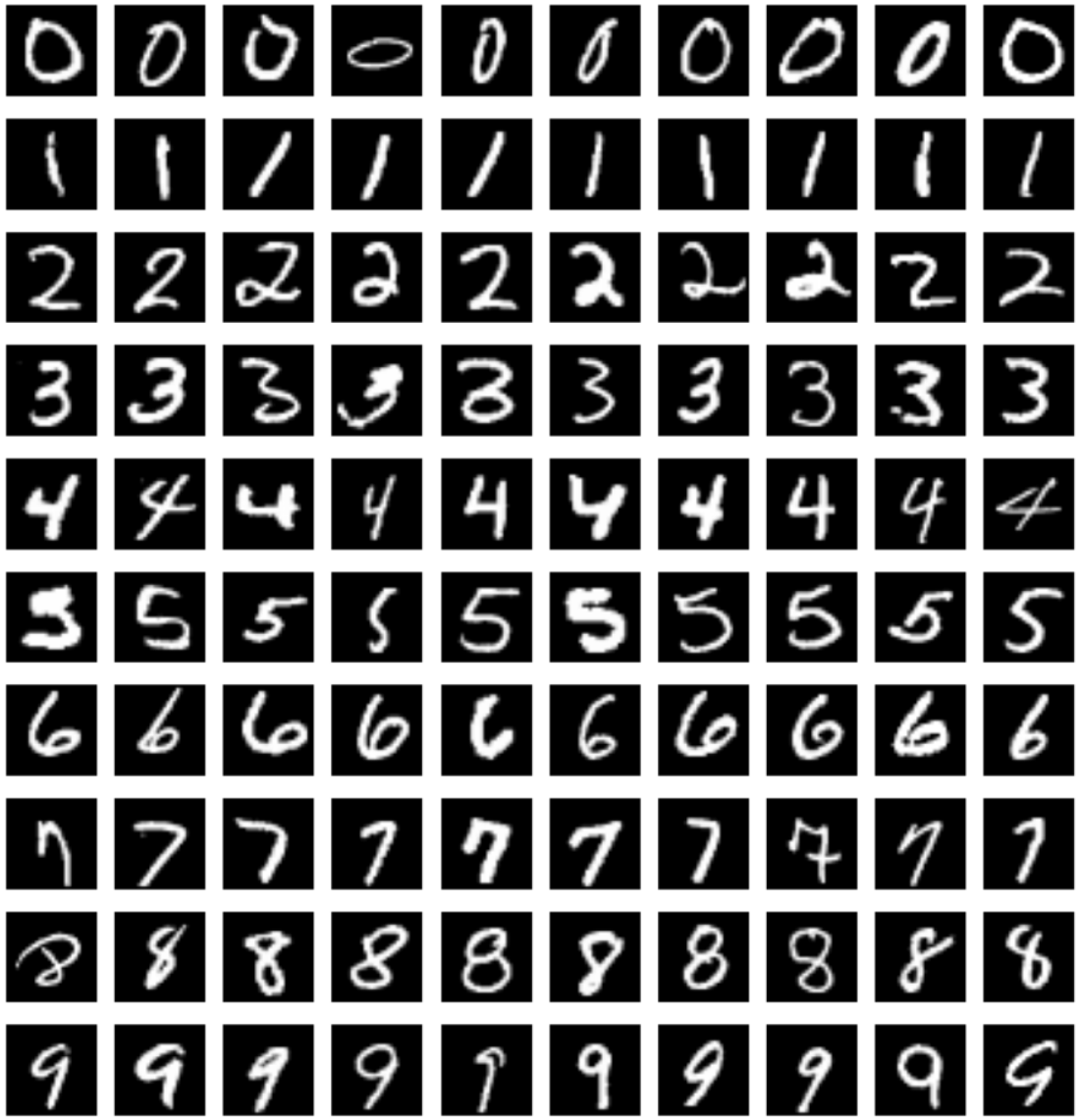
K-nn examples – MNIST

- Written digits
  - Training size: 60K

```
k=1, accuracy=97.22%
k=3, accuracy=97.11%
k=5, accuracy=96.96%
k=10, accuracy=96.55%
k=15, accuracy=96.48%
k=20, accuracy=96.18%
k=25, accuracy=95.92%
k=30, accuracy=95.75%
```

  - Training size: 5K

```
k=1, accuracy=93.61%
k=3, accuracy=93.12%
k=5, accuracy=93.08%
k=10, accuracy=92.37%
k=15, accuracy=91.89%
k=20, accuracy=91.45%
k=25, accuracy=90.93%
k=30, accuracy=90.46%
```



Colab link:
https://colab.research.google.com/drive/1EdDg6rk23BW7rO9uwhogJf5YSMsZ-zK4

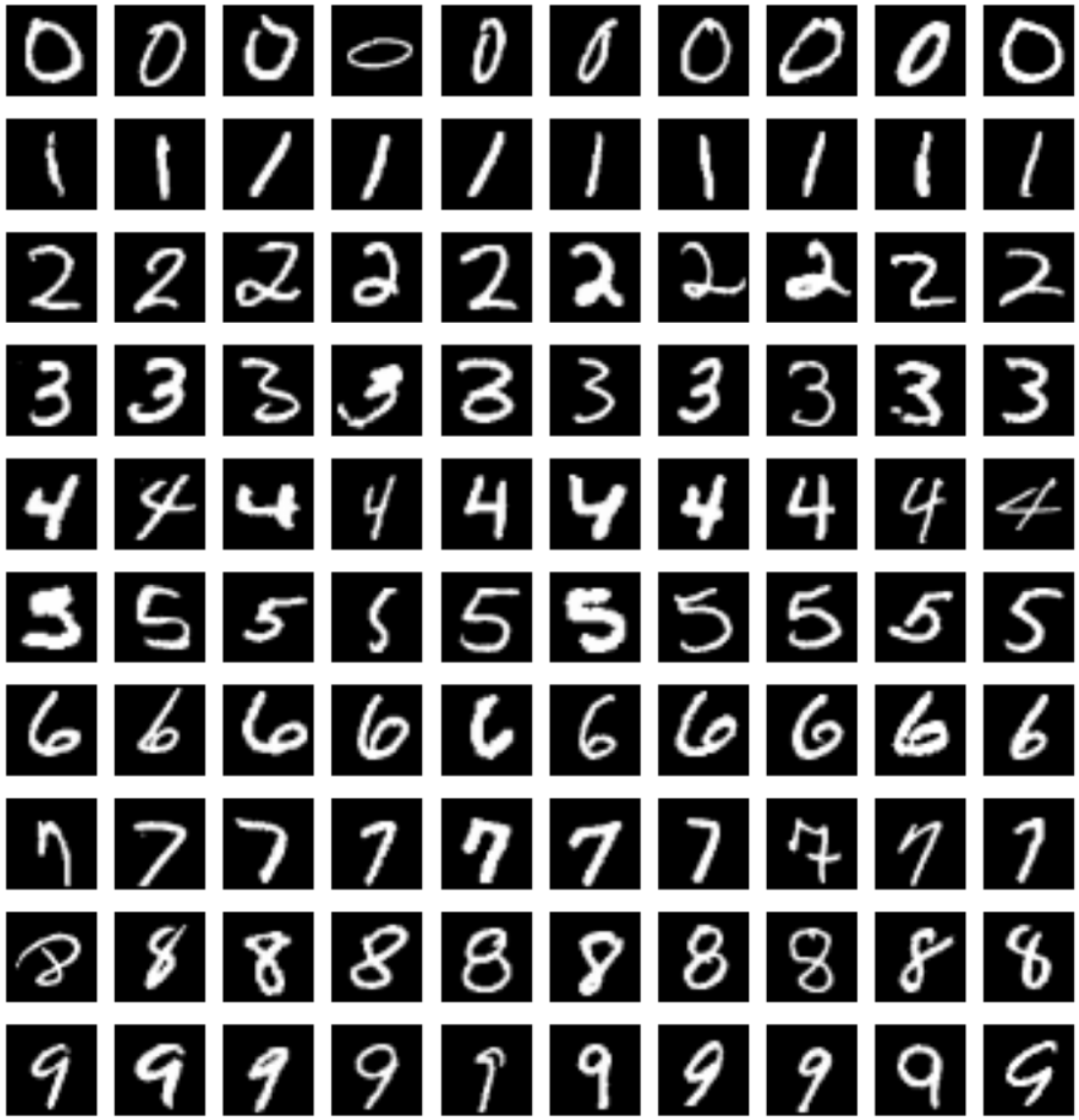
K-nn examples – MNIST

- Changing the split:
  - Training size: 60K

```
k=1, accuracy=97.16%
k=3, accuracy=97.31%
k=5, accuracy=97.22%
k=10, accuracy=96.69%
k=15, accuracy=96.60%
k=20, accuracy=96.36%
k=25, accuracy=96.04%
k=30, accuracy=95.78%
```

  - Training size: 5K

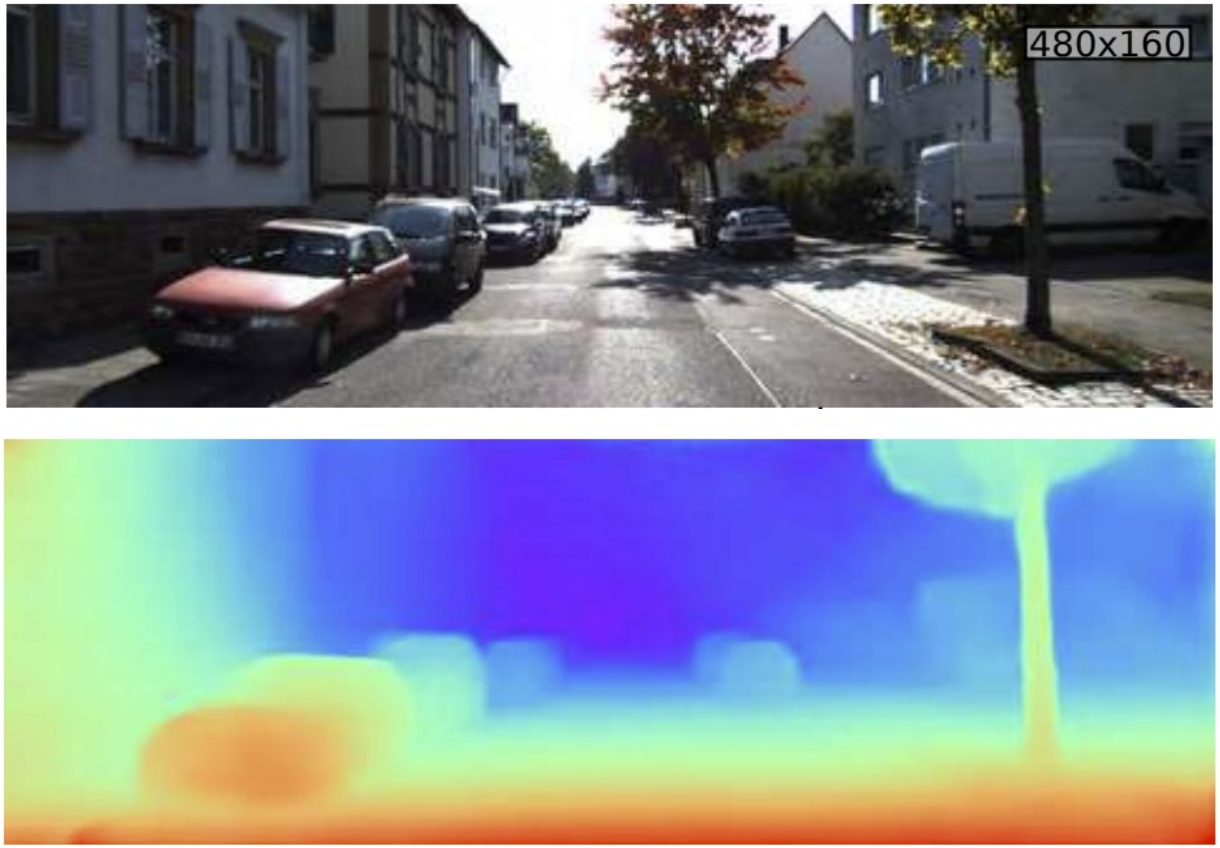
```
k=1, accuracy=93.10%
k=3, accuracy=92.98%
k=5, accuracy=92.68%
k=10, accuracy=92.14%
k=15, accuracy=91.60%
k=20, accuracy=90.90%
k=25, accuracy=90.37%
k=30, accuracy=89.97%
```



Colab link:
https://colab.research.google.com/drive/1EdDg6rk23BW7rO9uwhogJf5YSMsZ-zK4

Regression

- Learns a function that relates an input vector with a **continuous output**.
- Examples: depth prediction, weather forecasting, time-to-destination estimation…



Linear Regression

For more details, check Murphy chapter 11, Deisenroth chapter 9

- The input/output relation is linear in the model parameters w

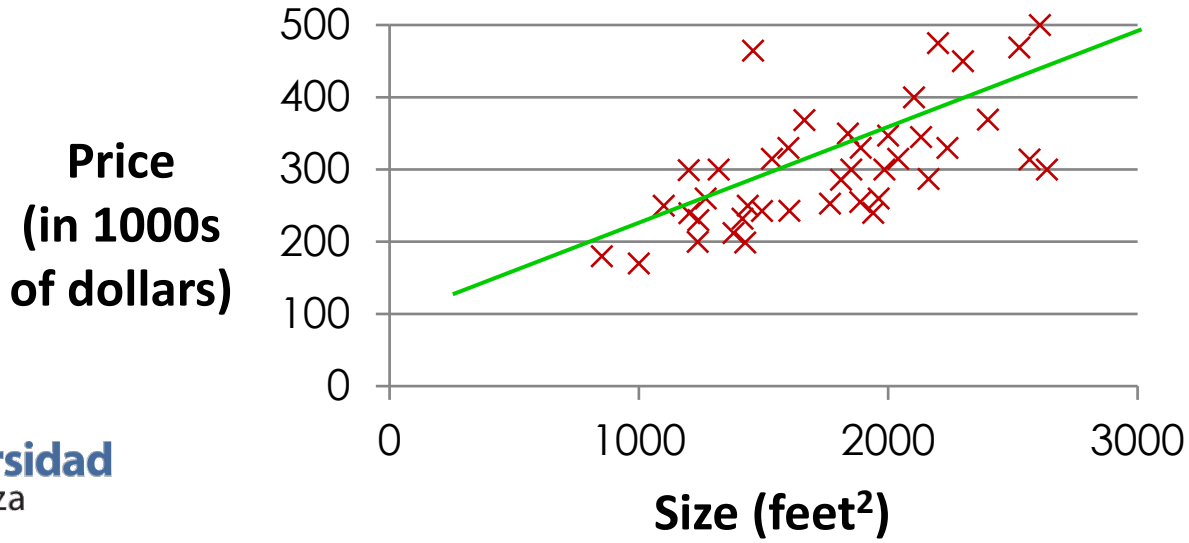
$$y = w^T x + \epsilon = \tilde{w}^T \tilde{x} + w_0 + \epsilon = \sum_{j=1}^{d} w_j x_j + \epsilon, \epsilon \sim N(0, \sigma^2)$$

$x = (1, x_1, \cdots, x_d)^T \rightarrow$ independent/explanatory variables

$w = (w_0, w_1, \cdots, w_d)^T$, $w_0 \rightarrow$ offset/bias, $w_{1:d} \rightarrow$ weights

$\tilde{x} = (x_1, \cdots, x_d)^T$, $\tilde{w} = (w_1, \cdots, w_d)^T$

- Example: Housing prices and size



How do we get w from data?

- Training data $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(i)}, y^{(i)}), \cdots, (x^{(N)}, y^{(N)})\}$
- Bayes' rule: $p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})}$
- MAP estimation:

$$\hat{w} = \arg max_w \, p(w|\mathcal{D}) = \arg max_w \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})} = \arg max_w \, p(\mathcal{D}|w)p(w)$$

- MLE estimation (we do not have prior knowledge over w):

$$\hat{w} = \arg max_w \, p(\mathcal{D}|w) = \arg max_w \; p(y|x, w) = \arg max_w \log(p(y|x, w))$$

- If the training samples are iid:

$$\hat{w} = \arg max_w \log\big(p(y|x, w)\big) = \arg max_w \log\left(\prod_i p\big(y^{(i)}|x^{(i)}, w\big)\right)$$

$$= \arg max_w \sum_i \log\Big(p\big(y^{(i)}|x^{(i)}, w\big)\Big)$$

How do we get w?

- As $\epsilon \sim N(0, \sigma^2)$

$$\arg max_w \sum_i \log\left(p\left(y^{(i)} \middle| x^{(i)}, w\right)\right) =$$

$$= \arg max_w \sum_i \log\left(\exp\left(-\frac{1}{2\sigma^2}\left(y^{(i)} - w^T x^{(i)}\right)^2\right)\right) =$$

$$= \arg min_w \sum_i \left(y^{(i)} - w^T x^{(i)}\right)^2 = \arg min_w \sum_i r^{(i)^2}$$

How do we get w?

- We can define then a cost function

$$J(w) = \frac{1}{2}\sum_i \left(y^{(i)} - w^T x^{(i)} \right)^2 = \frac{1}{2}\sum_i r^{(i)2}$$

- Our estimation $\hat{w}$ is the one minimizing the cost function

$$\hat{w} = \arg min_w J(w)$$

- Analytical solution for linear regression (ordinary least squares)

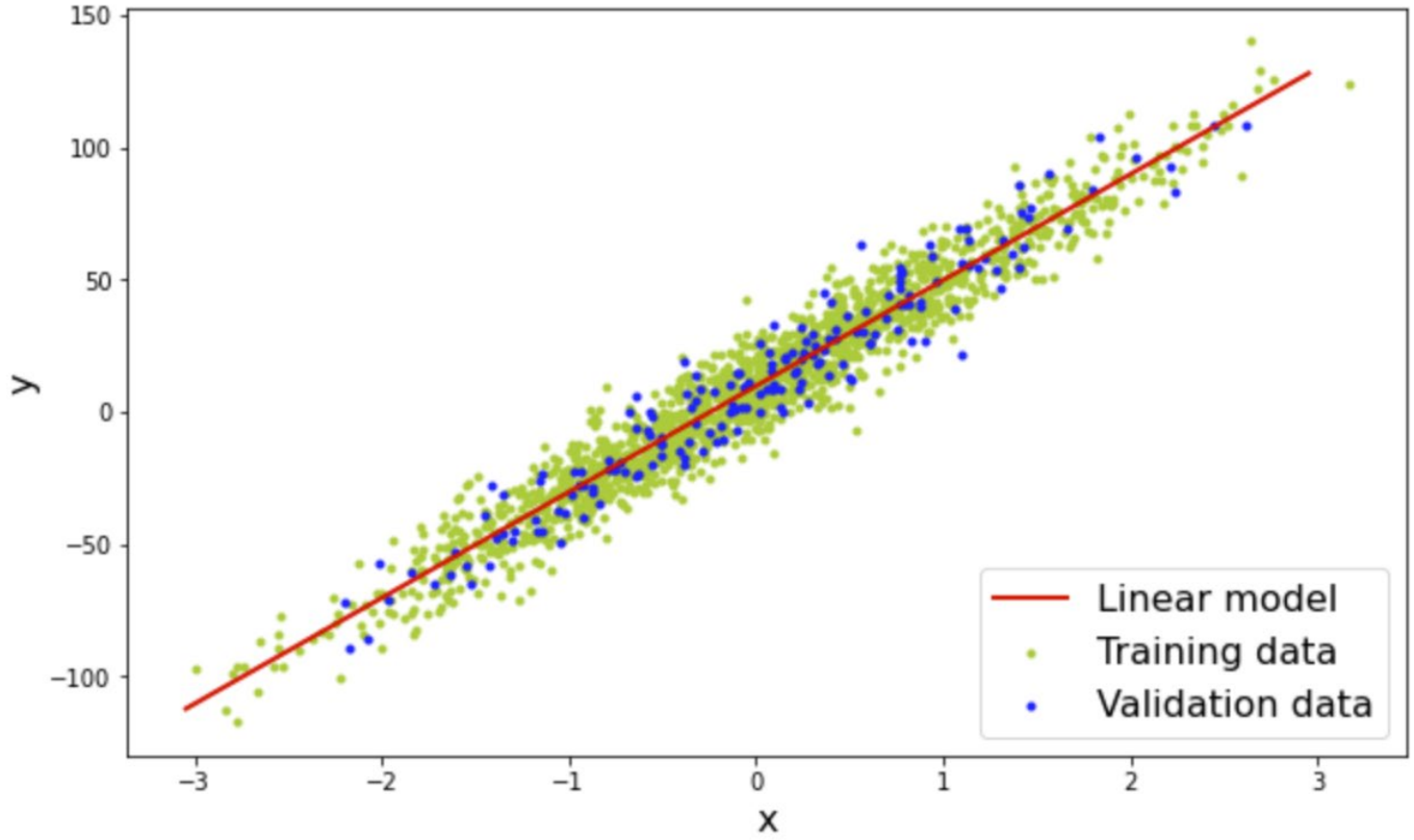
$$J(w) = \frac{1}{2}\sum_i \left(y^{(i)} - w^T x^{(i)} \right)^2 = \frac{1}{2}(Xw - y)^T(Xw - y)$$

$$\rightarrow \cdots \rightarrow \hat{w} = (X^T X)^{-1} X^T y$$

$$X = \begin{pmatrix} x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \cdots & x_d^{(N)} \end{pmatrix}, y = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{pmatrix}$$

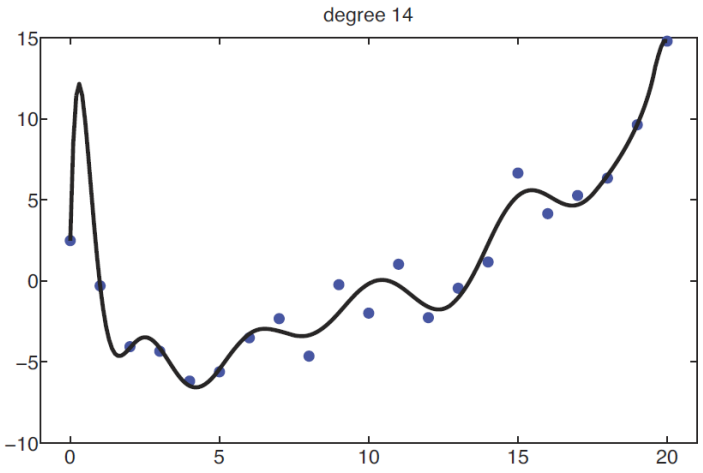
Simple linear regression example

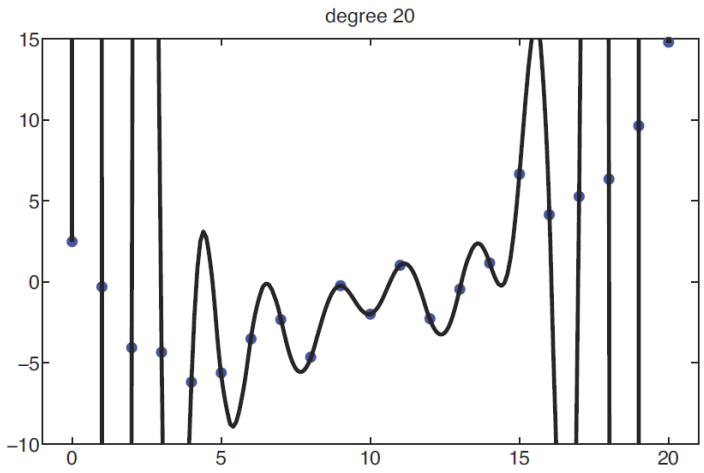
- Colab link: https://colab.research.google.com/drive/1EXDJHTIvp4zK3qavGsXO03NMwYRr59JM



Nonlinear Regression (linear in w_i!!)

- Basis function expansión
  - Linear regression can be made to model non-linear relations by replacing $x \rightarrow \phi(x)$
- Example:
  - Polinomial basis function $\phi(x) = \left[1, x, x^2, \cdots, x^d\right]^T$ $(x \in \mathbb{R}^1, \phi(x) \in \mathbb{R}^d)$
  - $y = w^T\phi(x) + \epsilon = \sum_{j=1}^{d} w_j x^j + \epsilon = w_0 + w_1 x^1 + w_2 x^2 + \cdots + w_d x^d + \epsilon$
  - Polynomia of degree 14 and 20 fit by LS (which one is better?)

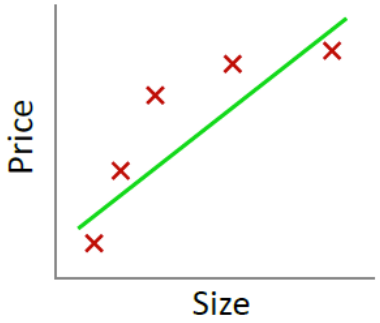




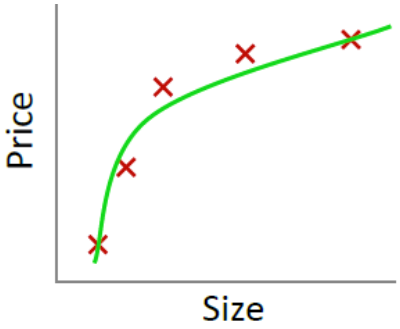
Example and figures taken from Murphy's book

Overfitting

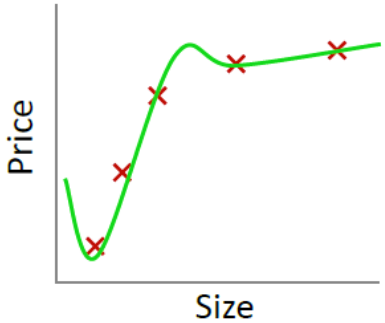
- The function $w^T\phi(x)$ fits the noise in addition to the underlying patterns. It "*memorizes*" the data instead of "*learning to generalize*".
- The function does not capture the structure of the data.
- The predictive capability of an overfitted model is poor.
- Techniques:
  - Cross-validation
  - Regularization



$\theta_0 + \theta_1 x$



$\theta_0 + \theta_1 x + \theta_2 x^2$

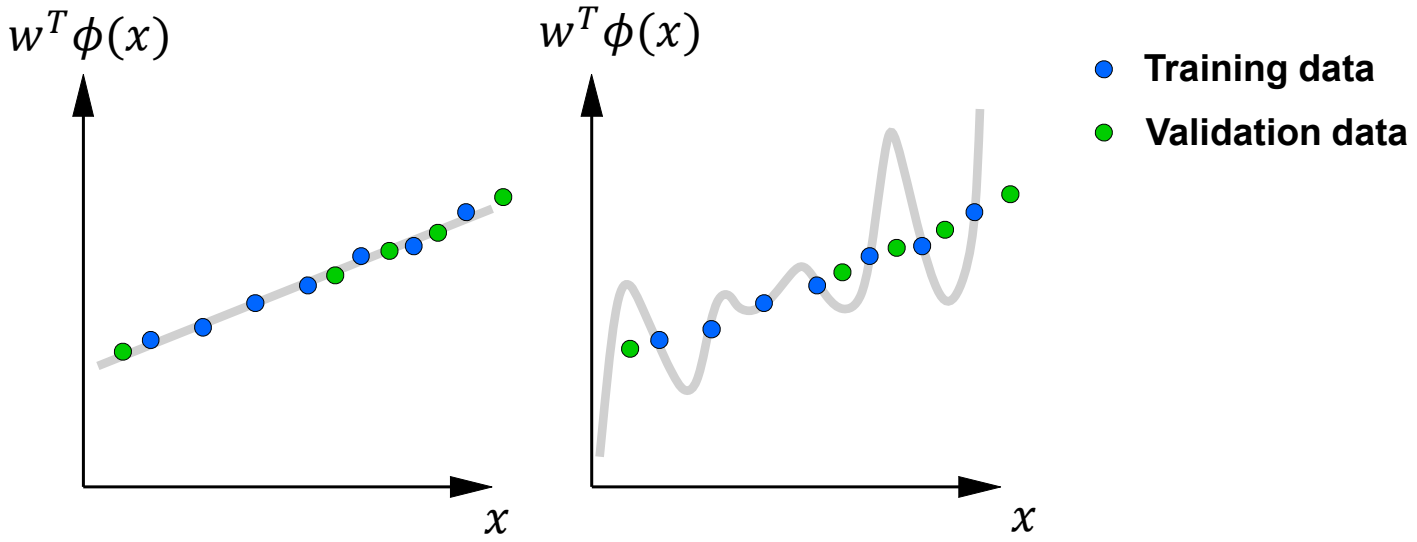


$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{N}(h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$$

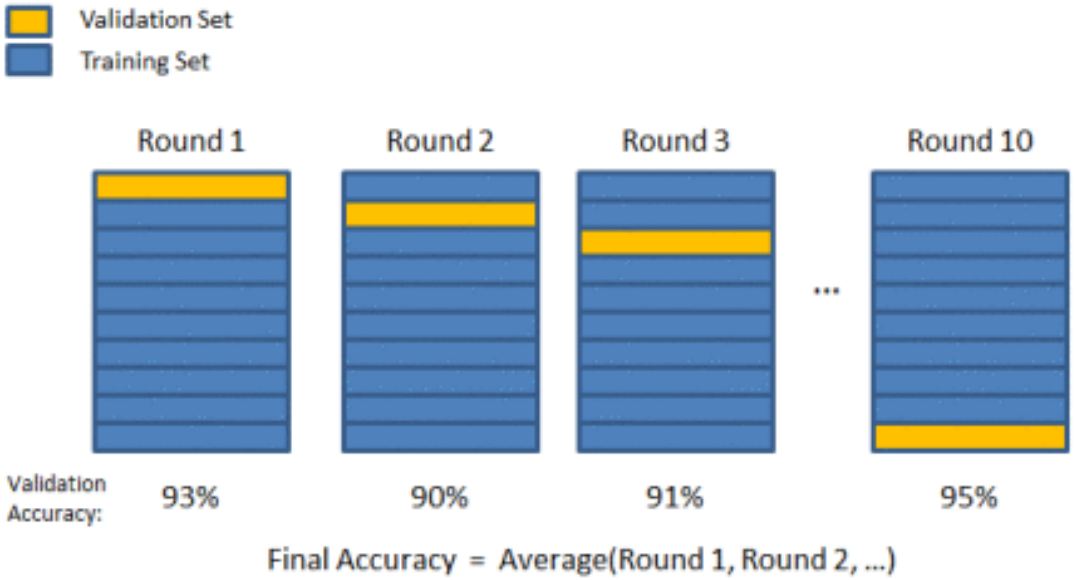
Cross-validation

- The data is partitioned into two sets.
  - The **training set** is used to estimate the model parameters w
  - The **validation set** is used to decide among a set of hyperparameters (models)
  - What we care about is generalization! → Final performance metrics should be given for a **test set**



K-fold cross-validation

- Common practice to have a higher degree of invariance to the validation/test split.
  - Separate the data into k folds
  - Do k training-validation rounds. A usual value for k is 10.
  - The validation/training sets are changed each round.
  - Metrics are averaged over all folds.



Summary: data splits

Model selection and fitting

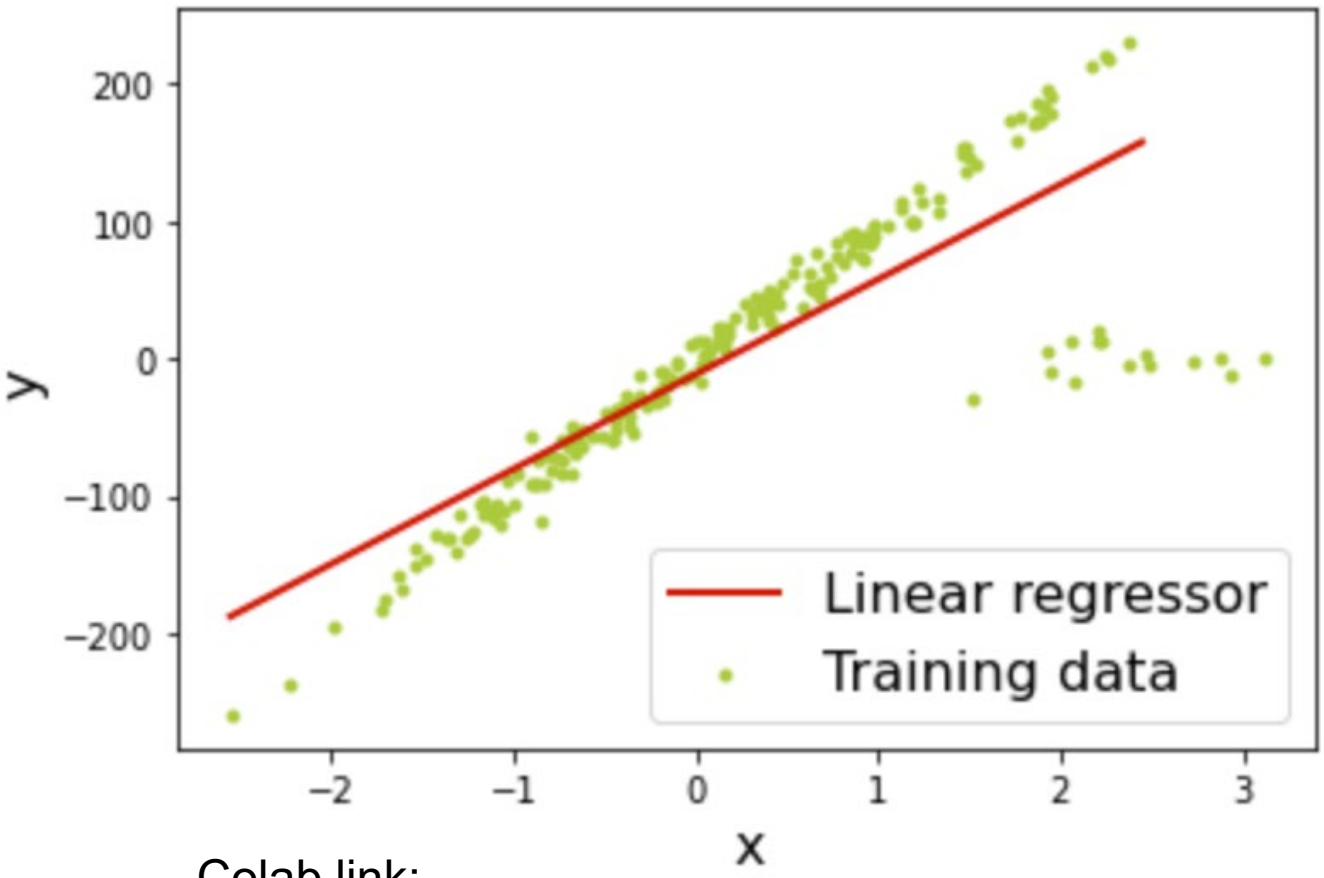
- Training data (~80% of the data)
  - Used to learn the model parameters
- Validation (~10% of the data)
  - Used to select the hyper-parameters
  - After used, can be added to the training data to train the final model
  - Training/validation splits can change during training (k-fold cross-validation)

Test characterization

- Test (~10% of the data)
  - Never used at the training stage!
  - Used to obtain the final specs of our model
  - After used, can be added to the training data to train the final model

Robust Regression

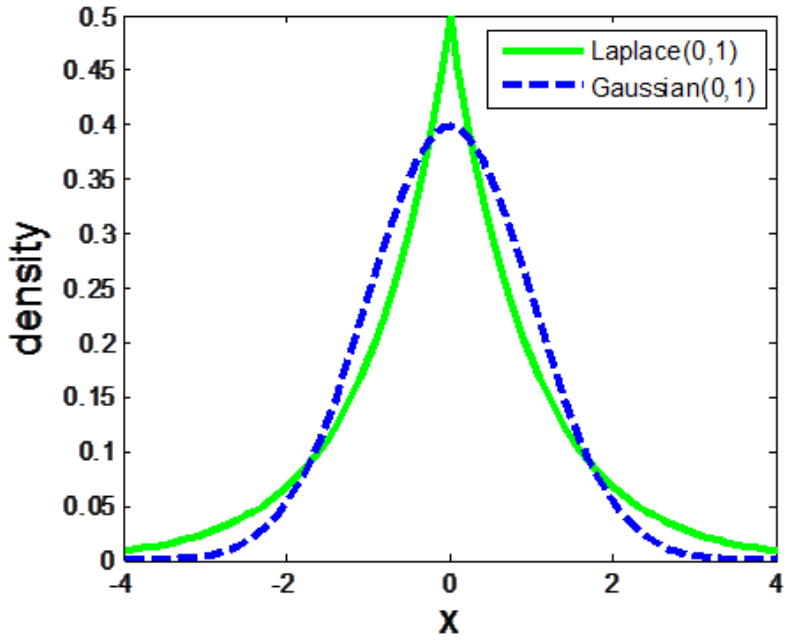
- The quadratic error penalty from the Gaussian noise gives a high weight to outliers



Colab link:
https://drive.google.com/file/d/1G5E6QHihfK1TyaYe6dxlF2FF5XC1kXn-

Robust Regression

- Probabilistic view: The low tolerance of Gaussian pdfs for high-noise data samples causes problems with outliers
- One possible solution: assume longer-tailed distributions, e.g., the Laplace one
- Other: detect outliers and remove them
  - You will learn about RANSAC in the computer vision course



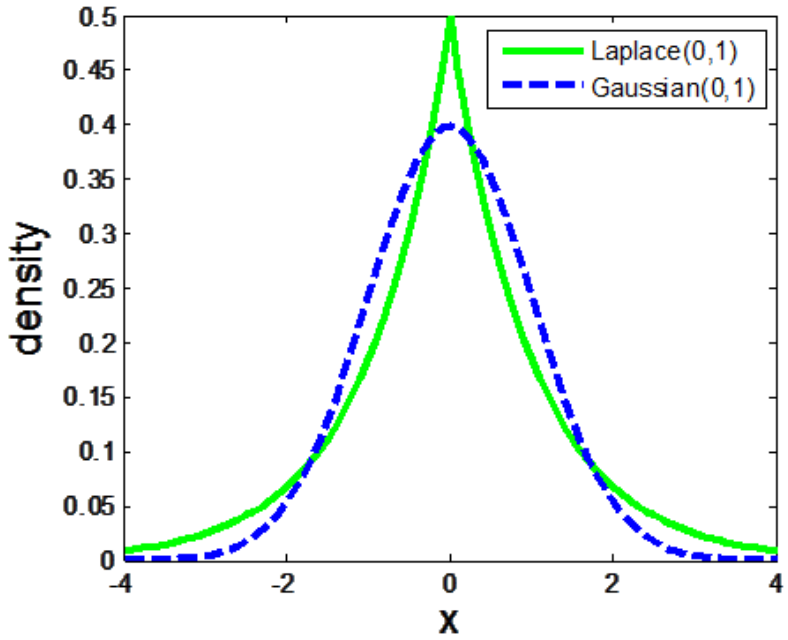
Robust Regression

- Laplace distribution:

$$p\left(y^{(i)}\middle|x^{(i)},w\right)=\exp\left(-\frac{1}{2b^2}\left|y^{(i)}-w^Tx^{(i)}\right|\right)\rightarrow\cdots\rightarrow$$

$$J(w)=\frac{1}{2}\sum_i\left|y^{(i)}-w^Tx^{(i)}\right|=\sum_i\left|r^{(i)}\right|$$

- Non-linear, hard to optimize!



Robust Regression: Huber loss function

- Equivalent to L2 near zero, to L1 for large errors

$$r^{(i)} = \begin{cases} \frac{r^{(i)^2}}{2} & if\ |r| \leq \delta \\ \delta|r| - \frac{\delta^2}{2} & if\ |r| > \delta \end{cases}$$

- Differentiable!

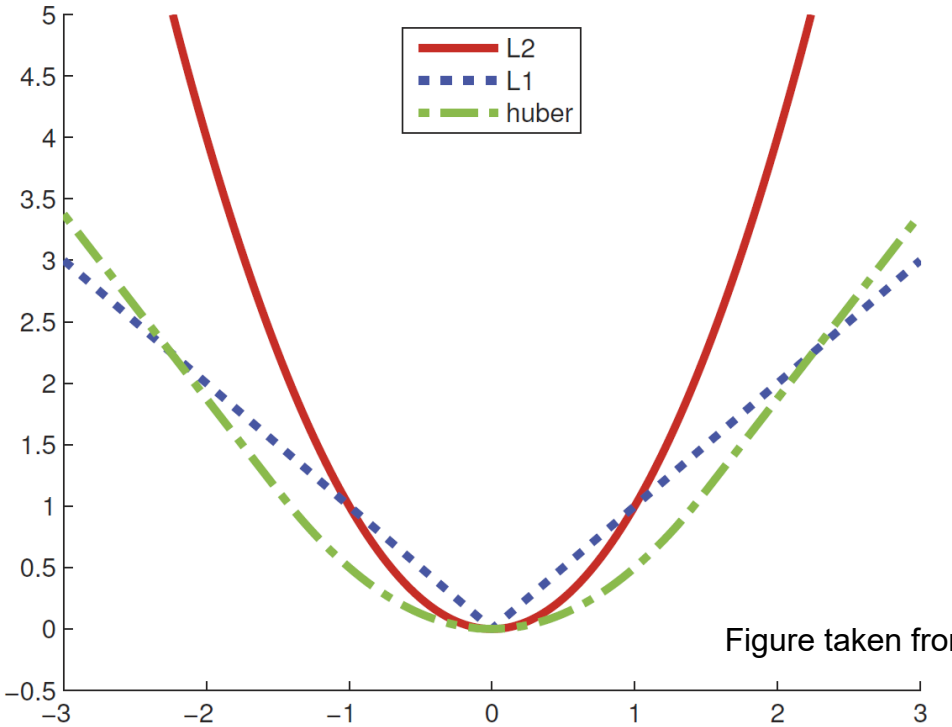
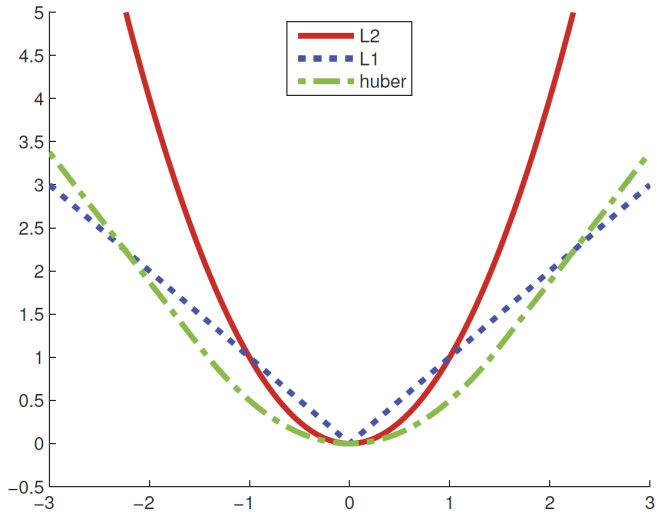


Figure taken from Murphy's book

Huber loss function

- How do we tune the parameter δ?

$$r^{(i)} = \begin{cases} \dfrac{{r^{(i)}}^2}{2} & if\ |r| \leq \delta \\ \delta|r| - \dfrac{\delta^2}{2} & if\ |r| > \delta \end{cases}$$

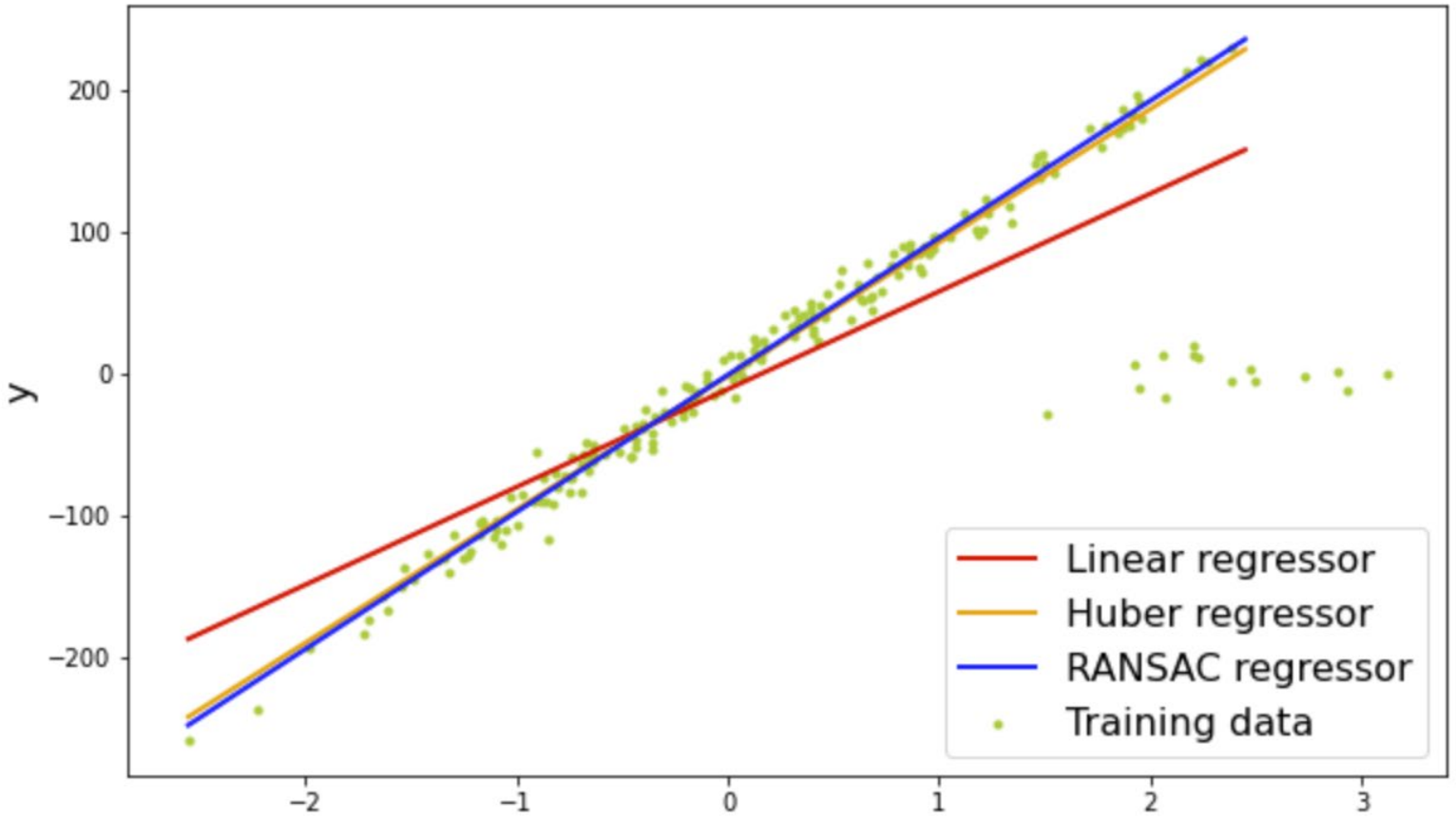


- δ is the "boundary" between inliers (Gaussian pdf assumed) and the outliers (Laplacian pdf assumed). So:
  1. [Huber 1981], says: "good choices are between σ and 2σ"
     - σ is the standard deviation of the residuals $r^{(i)}$ of the inliers
  2. If you do not know σ, you estimate it from the training data residuals $r^{(i)}$.
     - IMPORTANT! You have outliers, so use a robust estimator for σ. MAD (Median Absolute Deviation) is common:

$$\hat{\sigma} = 1.4826 \cdot \text{MAD}, \text{MAD} = \text{median}\left(\left|r^{(1)} - \tilde{r}\right|, \ldots, \left|r^{(N)} - \tilde{r}\right|\right), \tilde{r} = \text{median}\left(r^{(1)}, \ldots, r^{(N)}\right)$$

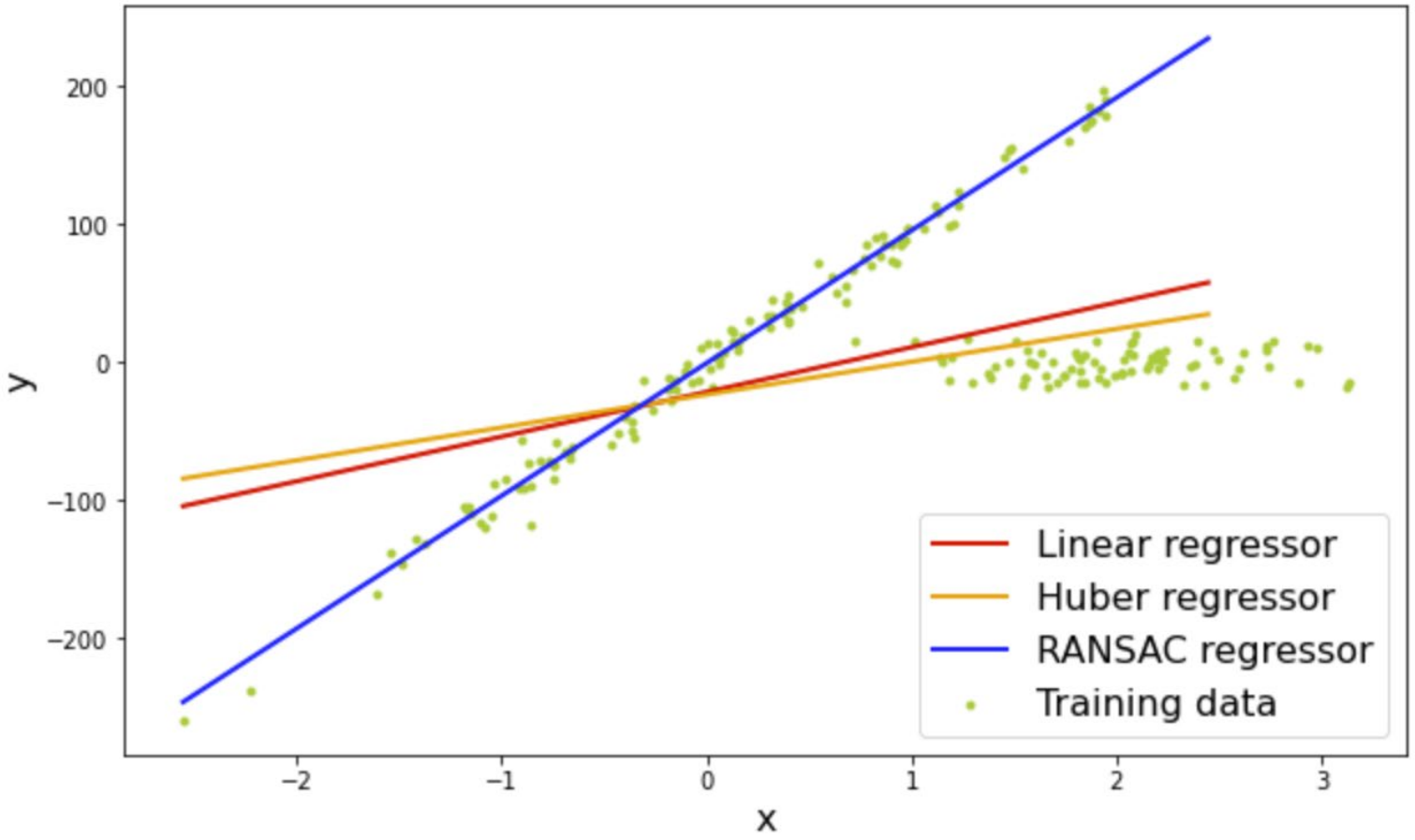
Robust regression example

- The Huber loss and RANSAC are robust to outliers



Robust regression example

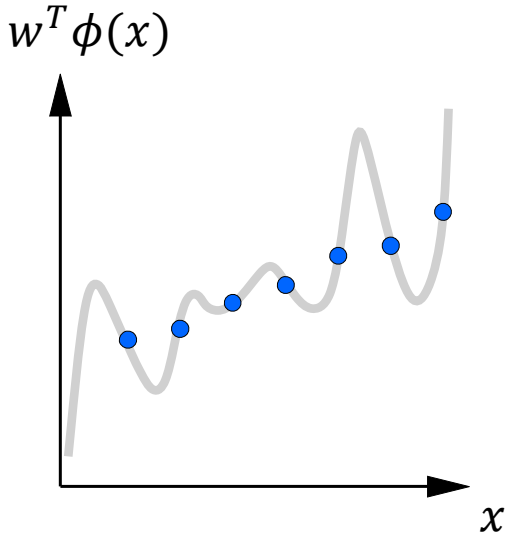
- But RANSAC can tolerate a higher outlier ratio!



Ridge Regression

- MLE can overfit easily for noisy training data
- Complex models will fit the noise
- Solution: Use a Gaussian prior over the model parameters (remember MAP from slide 23!)

$$p(w) \sim N(0, \tau^2)$$



Ridge Regression

- The MAP estimation is

$$\hat{w} = \arg max_w \, p(\mathcal{D}|w)p(w) =$$
$$= \arg max_w \sum_i \log\left(\exp\left(-\frac{1}{2\sigma^2}\left(y^{(i)} - w^T x^{(i)}\right)^2\right)\right) + \sum_j \log\left(\exp\left(-\frac{1}{2\tau^2}\left(w_j\right)^2\right)\right)$$

- Resulting in the following cost function

$$J(w) = \frac{1}{N}\sum_i \left(y^{(i)} - w^T x^{(i)}\right)^2 + \lambda w^T w; \qquad \lambda = \frac{\sigma^2}{\tau^2}$$

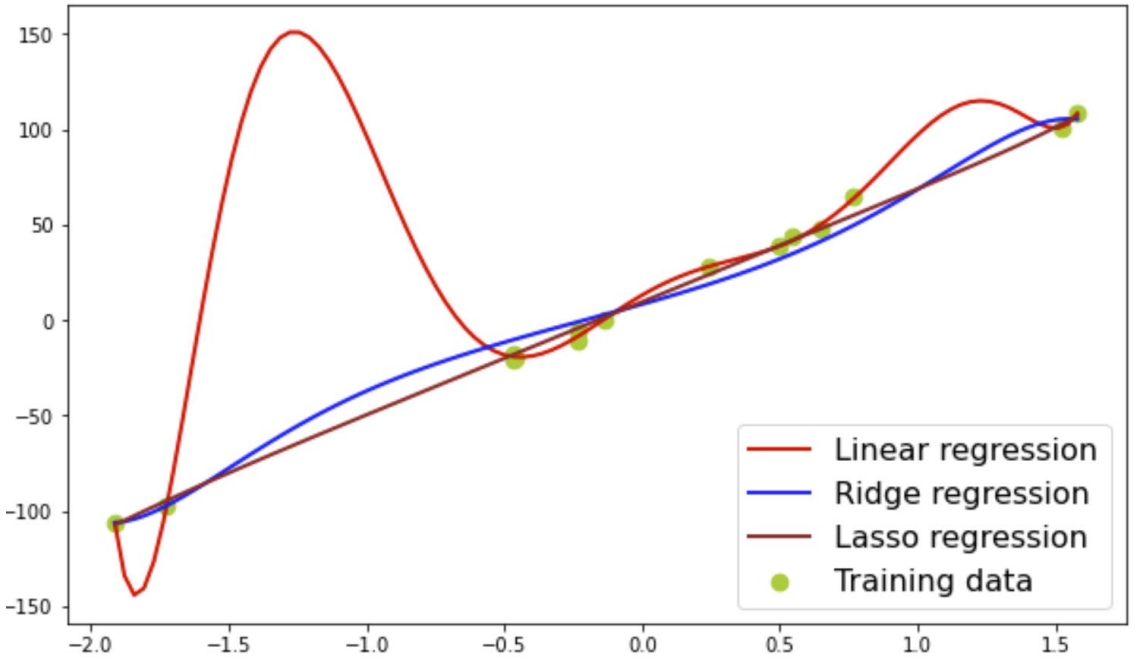
- Which has the following solution $\hat{w} = (\lambda I_d + X^T X)^{-1} X^T y$

- λ usually selected by cross-validation

Ridge Regression

Colab link:
https://drive.google.com/file/d/1EnrFqx8Ks0L3-Uy99IB6DJ4DdEHyP5pl

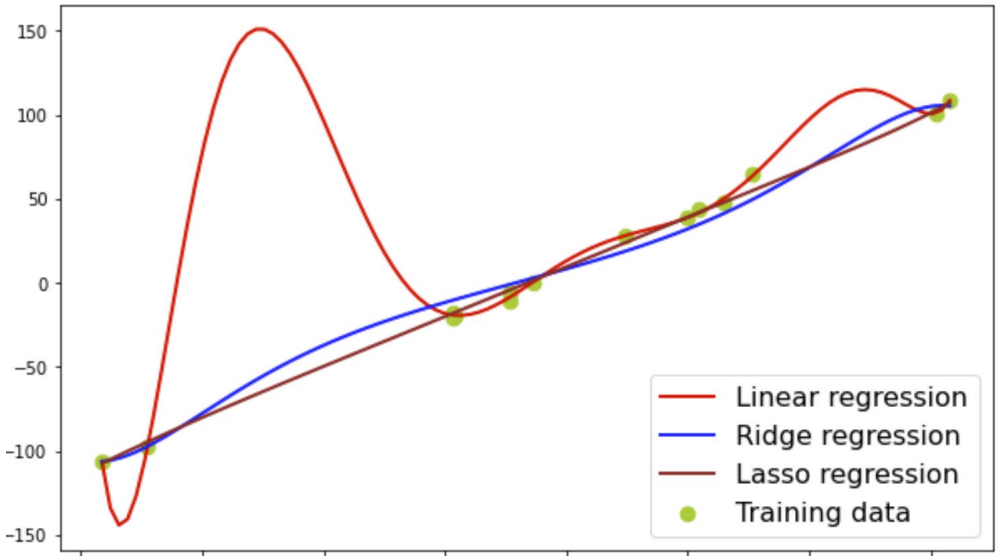
- Avoids overfitting
- $\lambda w^T w = \lambda \|w\|_2$ AKA "weight decay" in deep learning
- $\lambda \|w\|_1$ LASSO (Least Absolute Shrinkage and Selection Operator)
  - Assumes a Laplacian distribution for the weights (then $\lambda = \frac{2\sigma^2}{b^2}$)
  - Can assign zero values to some coefficients (feature selection)



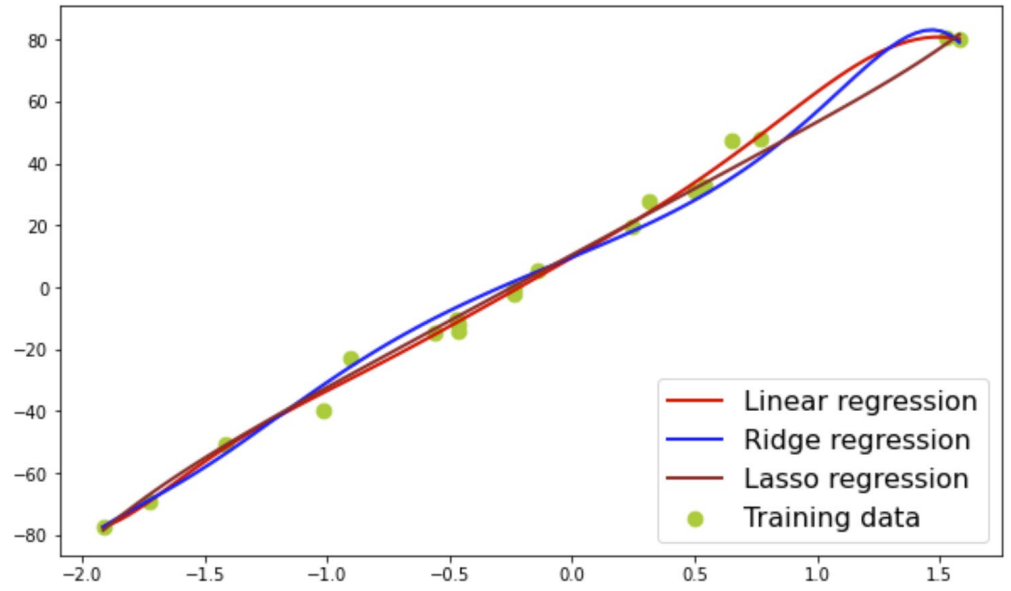
Data regularization

Colab link:
https://colab.research.google.com/drive/1j3kYQAHeGSbEFf66U_f2P9wlSatT1Erl

- BTW, in this example, check the regularization effect of adding more data
  - 15 data samples

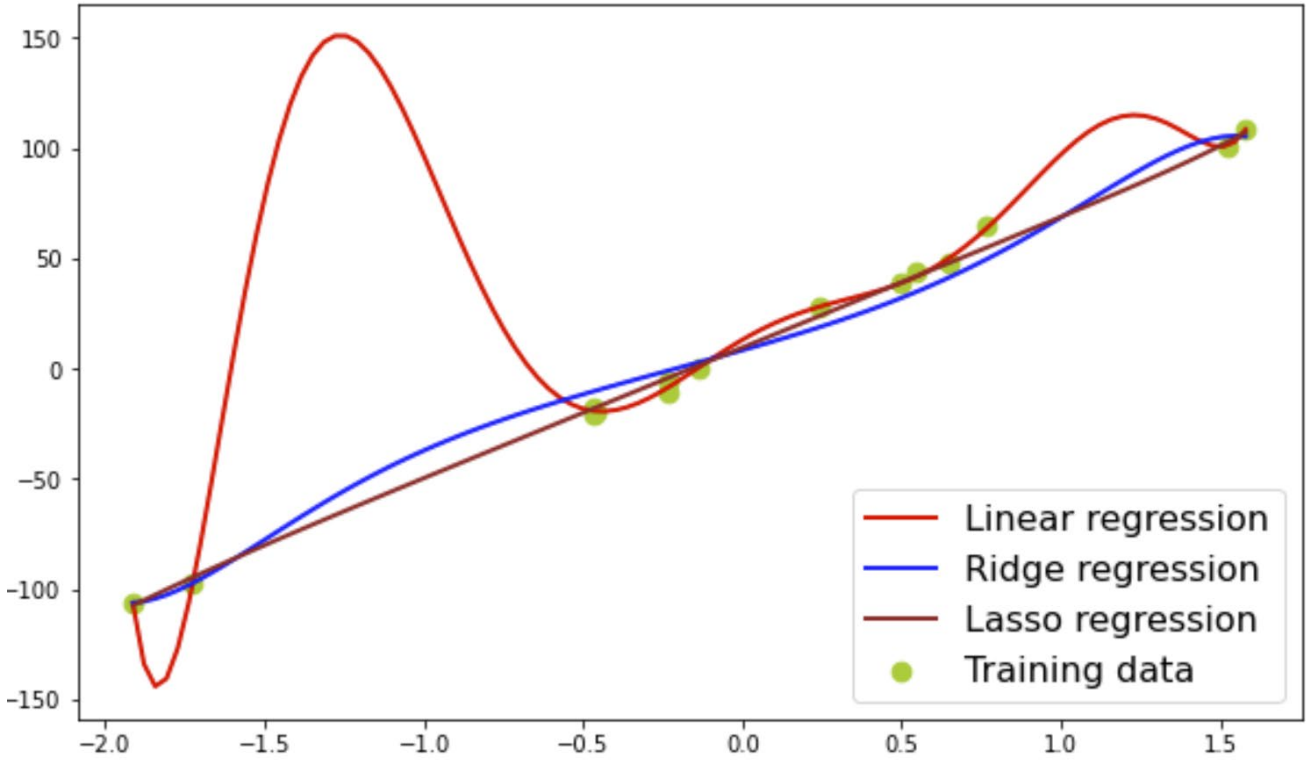


  - 20 data samples



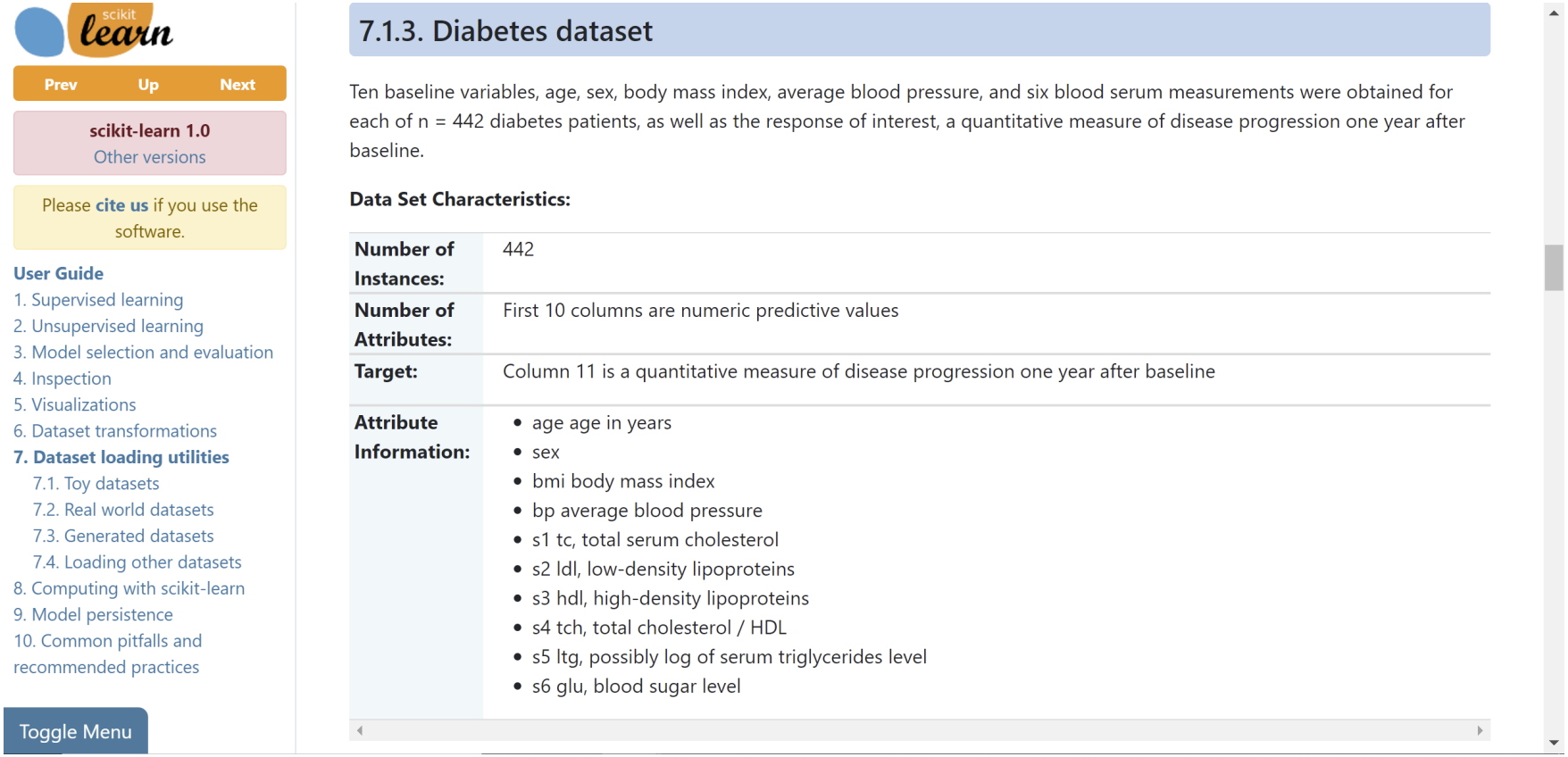
Summary: avoiding overfitting

- Do cross-validation to select the best model
- Add constraints on your model weights
- Add more training data!



Regression example: The diabetes dataset

- Colab link: https://drive.google.com/file/d/1EIKbBCWetqEVKumrTX4GEpxdbRSFXwVw



scikit learn

Prev Up Next

scikit-learn 1.0
Other versions

Please cite us if you use the software.

User Guide
1. Supervised learning
2. Unsupervised learning
3. Model selection and evaluation
4. Inspection
5. Visualizations
6. Dataset transformations
7. Dataset loading utilities
   7.1. Toy datasets
   7.2. Real world datasets
   7.3. Generated datasets
   7.4. Loading other datasets
8. Computing with scikit-learn
9. Model persistence
10. Common pitfalls and recommended practices

Toggle Menu

7.1.3. Diabetes dataset

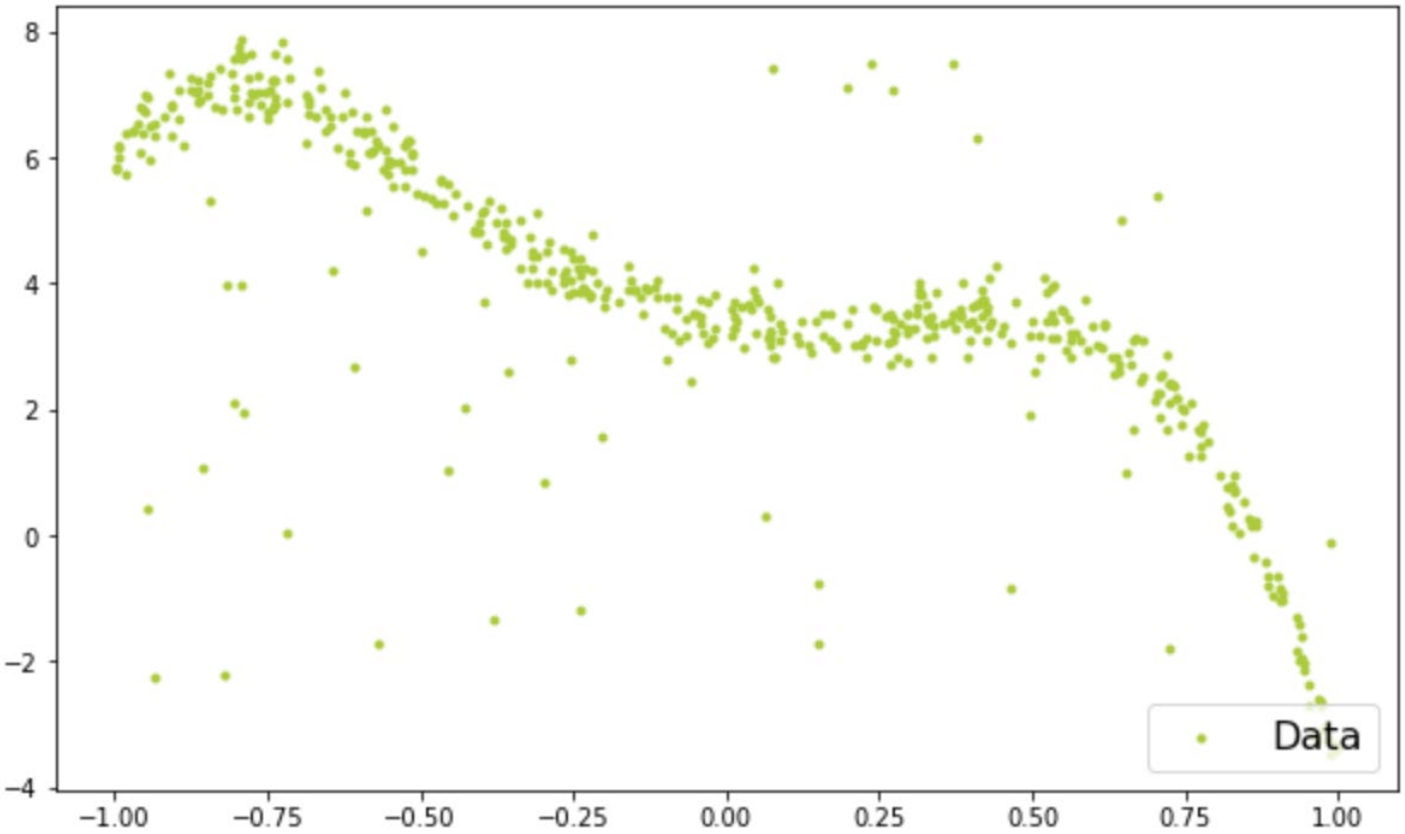
Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of n = 442 diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.

Data Set Characteristics:

Number of Instances:	442
Number of Attributes:	First 10 columns are numeric predictive values
Target:	Column 11 is a quantitative measure of disease progression one year after baseline
Attribute Information:	• age age in years • sex • bmi body mass index • bp average blood pressure • s1 tc, total serum cholesterol • s2 ldl, low-density lipoproteins • s3 hdl, high-density lipoproteins • s4 tch, total cholesterol / HDL • s5 ltg, possibly log of serum triglycerides level • s6 glu, blood sugar level

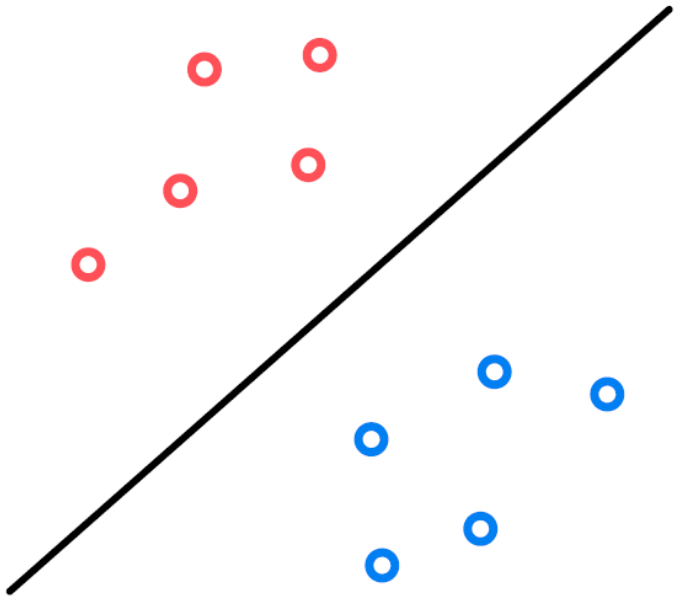
Exercise: Fit a polynomial model

- Link to Colab: https://drive.google.com/file/d/1G1N9C9Yry-3Lc9A1mL5yawM1rDB6TnKM
- Link to data: https://drive.google.com/file/d/1G_XseoB0lhII21jGdEkSYQyaEtMoSfQy



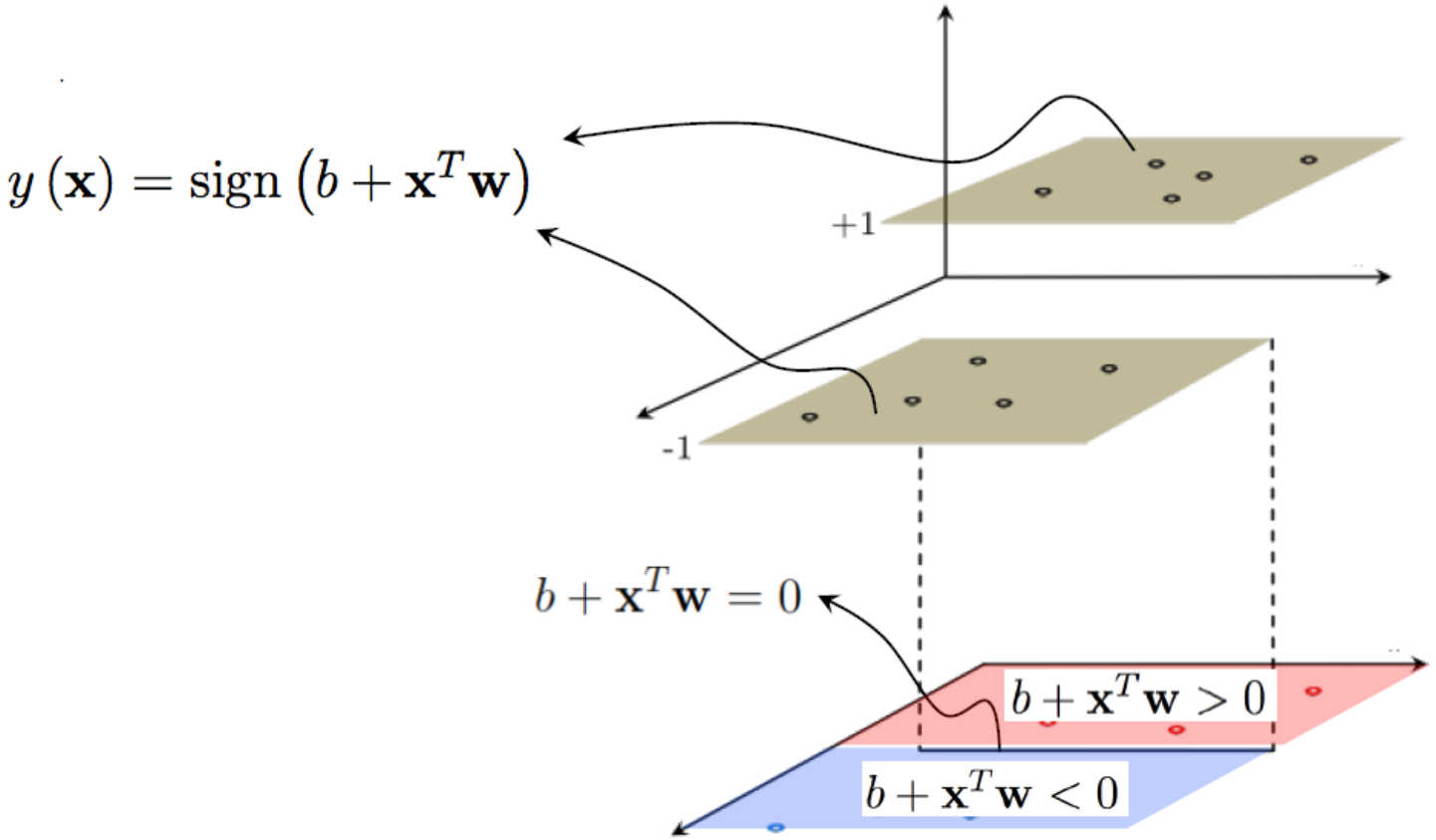
Linear Classification

- Classification: The output of the learned model is discrete
- Linear classification: The model is based on a linear combination of the input features
- Intuition: there is an hyperplane $\pi \in \mathbb{R}^{d-1}$ (e.g., a line for a 2D space) that separates the two classes/categories.



Binary Classification (intuition)

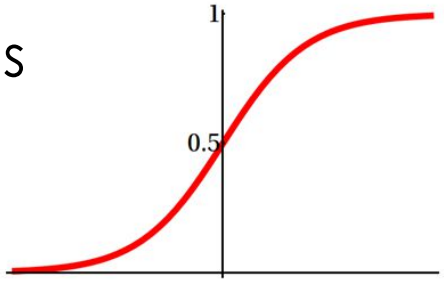
- We can get a discrete output $y \in \{-1, +1\}$ by taking the sign of the signed point-to-hyperplane distance
- The linear decision boundary is perpendicular to w
- The formulation can be extended easily to multi-class



Sigmoid and tanh (intuition)

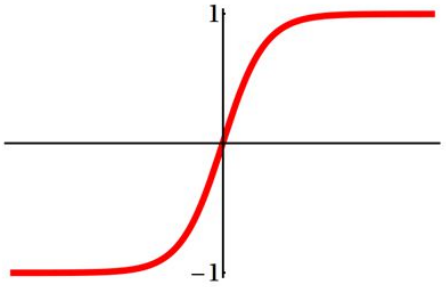
- The sign() function is not continuous.
- We can use continuous functions that approximate the sign function.
  - The sigmoid function
  - The tanh function

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

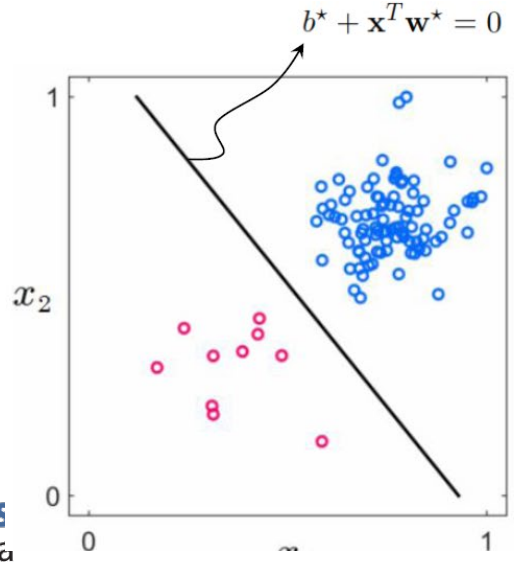


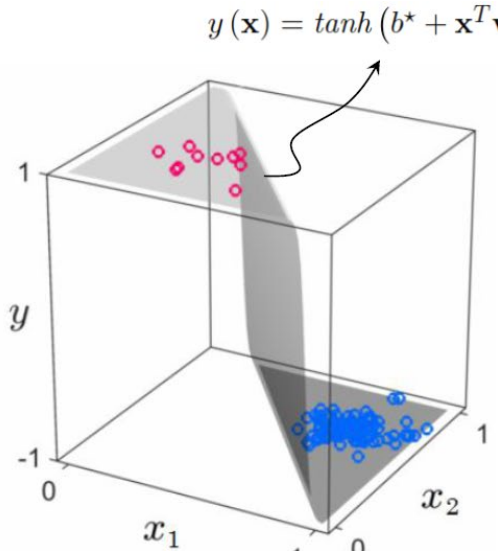
logistic (sigmoid, unipolar)

$$\tanh(\Sigma) = \frac{e^{\Sigma}-e^{-\Sigma}}{e^{\Sigma}+e^{-\Sigma}}$$



tanh (bipolar)





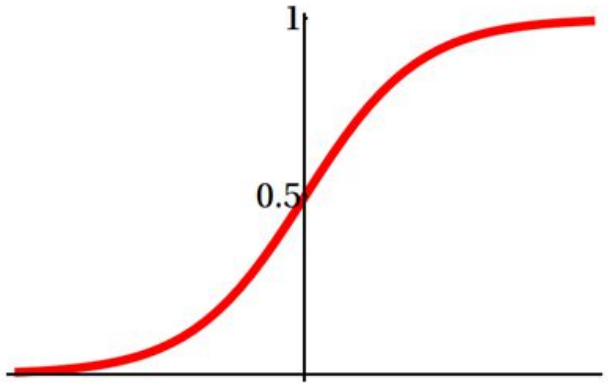
Logistic Regression (probabilistic interpretation)

- The input/output model is

$$p(y|x,w) = Ber(y|\sigma(w^T x)) \rightarrow$$
$$\begin{cases} p(y=1|x,w) = \sigma(w^T x) \\ p(y=0|x,w) = 1 - \sigma(w^T x) \end{cases}$$

  - $w^T x \rightarrow \infty \Longrightarrow \sigma(w^T x) \rightarrow 1 \Longrightarrow$
    $\Longrightarrow p(y=1|x,w) \rightarrow 1 \ \ (p(y=0|x,w) \rightarrow 0)$
  - $w^T x \rightarrow -\infty \Longrightarrow \sigma(w^T x) \rightarrow 0 \Longrightarrow$
    $\Longrightarrow p(y=1|x,w) \rightarrow 0 \ \ (p(y=0|x,w) \rightarrow 1)$
  - $w^T x = 0 \Longrightarrow \sigma(w^T x) \rightarrow 0.5 \Longrightarrow$
    $\Longrightarrow p(y=1|x,w) = 0.5 \ \ (p(y=0|x,w) \rightarrow 0.5)$

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



logistic (sigmoid, unipolar)

For more details, check Murphy chapter 10

How do we get w?

- Training data $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(i)}, y^{(i)}), \cdots, (x^{(N)}, y^{(N)})\}$
  - $x^{(i)} \in \mathbb{R}^d$, $y^{(i)} \in \begin{cases} \{0, 1\} \\ \{-1, +1\} \end{cases}$ (depends if we use sigmoid or tanh)
- Bayes' rule: $p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})}$
- MAP estimation (leads to L1/L2 norm regularization):

$$\hat{w} = \arg max_w \, p(w|\mathcal{D}) = \arg max_w \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})}$$
$$= \arg max_w \, p(\mathcal{D}|w)p(w)$$

- MLE estimation (we do not have prior knowledge over w):

$$\hat{w} = \arg max_w \, p(\mathcal{D}|w) = \arg max_w \; p(y|x, w)$$
$$= \arg max_w \log(p(y|x, w))$$

How do we get w?

- If the training samples are iid:

$$\hat{w} = \arg max_w \log\big(p(y|x,w)\big) = \arg max_w \log\left(\prod_i p\big(y^{(i)}\big|x^{(i)},w\big)\right)$$

$$= \arg min_w - \sum_i \log\Big(p\big(y^{(i)}\big|x^{(i)},w\big)\Big)$$

- $p\big(y^{(i)}\big|x^{(i)},w\big)$ is the one described before, resulting in

$$\hat{w} = \arg min_w - \sum_i \log\Big(p\big(y^{(i)}\big|x^{(i)},w\big)\Big) = \ldots =$$

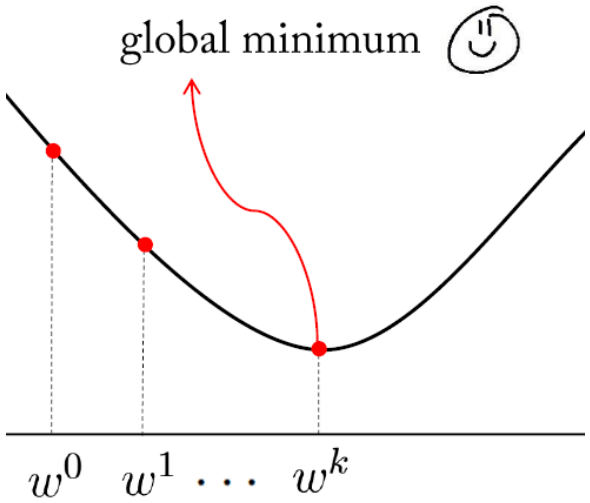
$$= \begin{cases} -\sum_i \big(y^{(i)} \log\big(w^T x^{(i)}\big) + (1-y^{(i)}) \log\big(1 - w^T x^{(i)}\big)\big), if\ y^{(i)} \in \{0,1\} \\ -\sum_i \big(\log\big(1+\exp\big(-y^{(i)} w^T x^{(i)}\big)\big)\big), if\ y^{(i)} \in \{-1,+1\} \end{cases}$$

Cost function (loss)

- Cross-entropy loss: $L =$

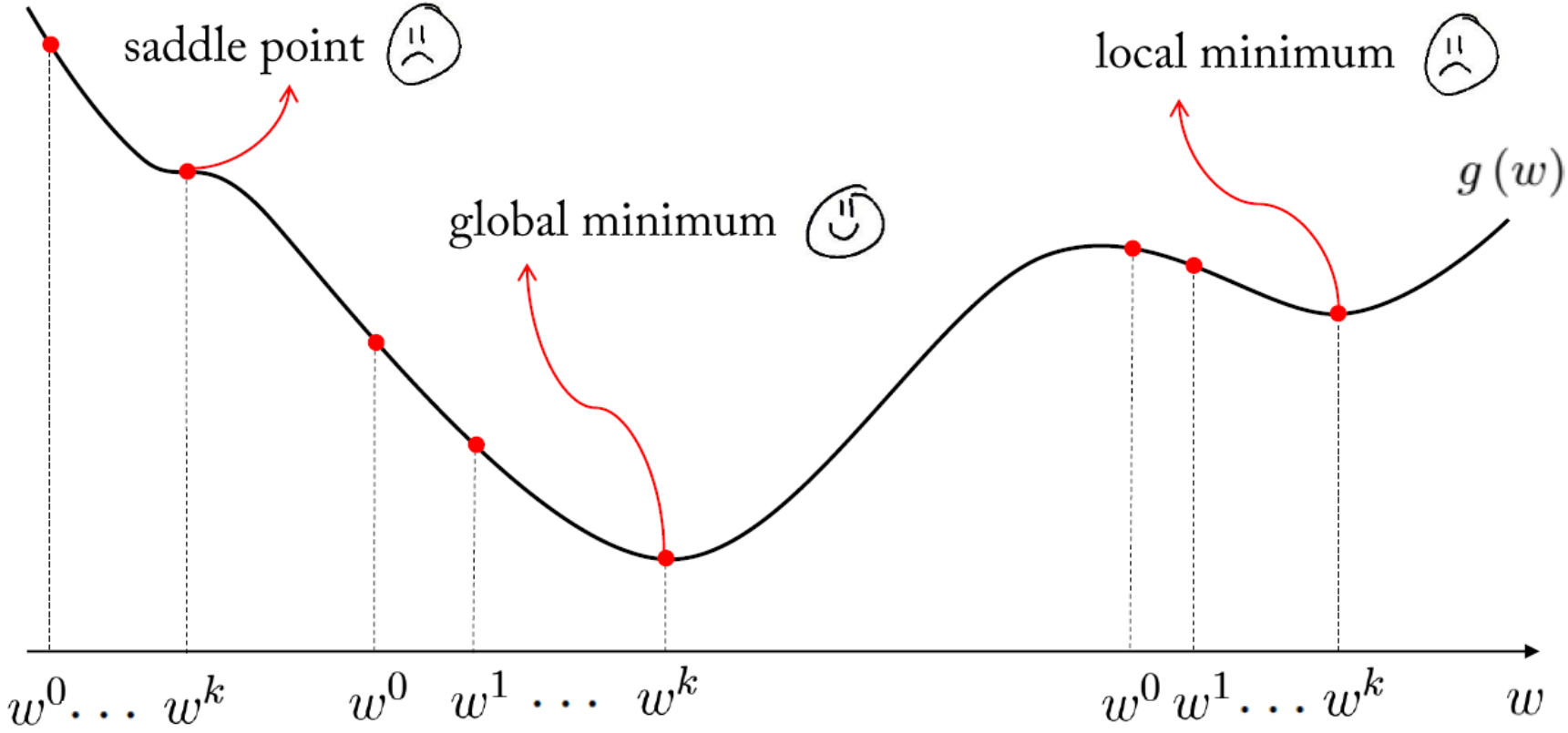
$$\begin{cases} -\sum_i \left(y^{(i)} \log\left(w^T x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - w^T x^{(i)}\right)\right), if\ y^{(i)} \in \{0, 1\} \\ \sum_i \left(\log\left(1 + \exp\left(-y^{(i)} w^T x^{(i)}\right)\right)\right), if\ y^{(i)} \in \{-1, +1\} \end{cases}$$

- Differently from linear regression, no closed-form solution!...
- … but, the loss is convex and has a single minimum (gradient descent will take us to it)



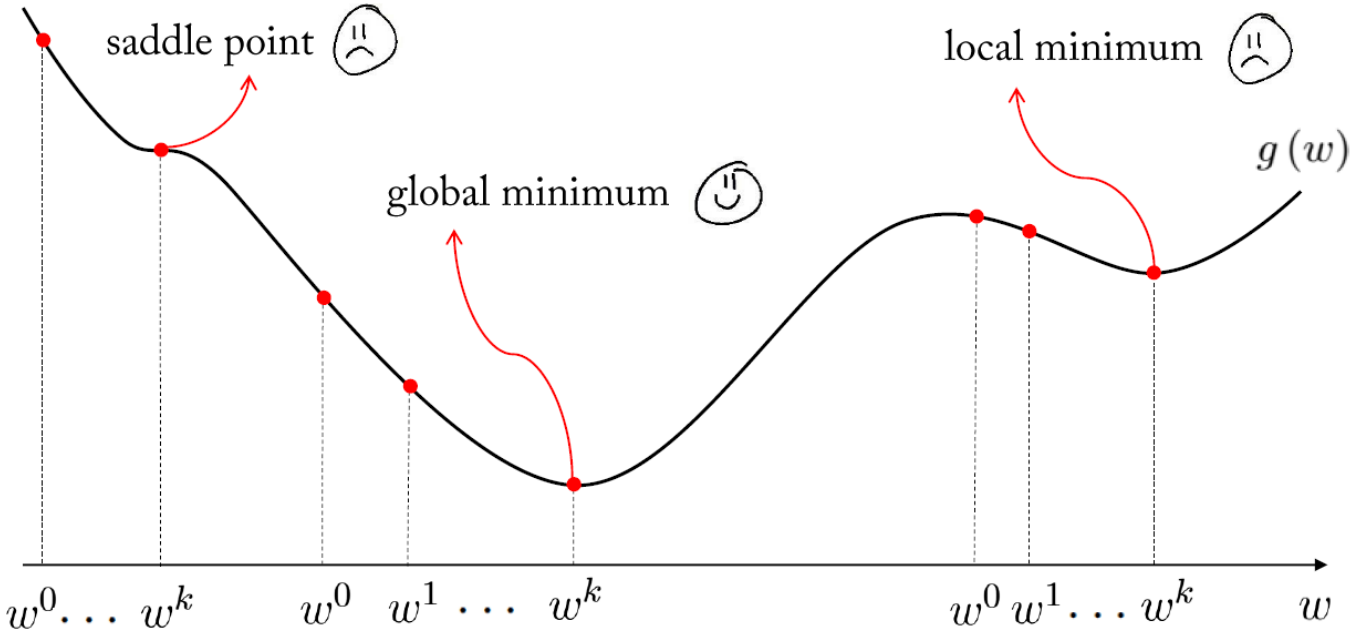
Gradient descent

- Gradient descent → Iterative weight updates in the downward direction, $w_{k+1} = w_k - \eta_k \frac{dg(w)}{d(w)}$
- In general, it converges to a local minimum



Gradient descent

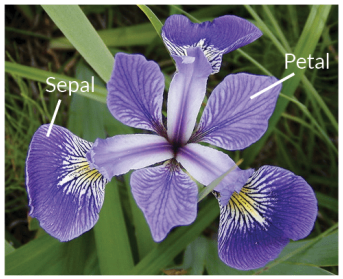
- Issues:
  - Select the initial seed w_0
  - Select the learning rate η_k
    - Small $\eta_k \rightarrow$ slow convergence, large $\eta_k \rightarrow$ fail to converge



Logistic regression in the Iris dataset

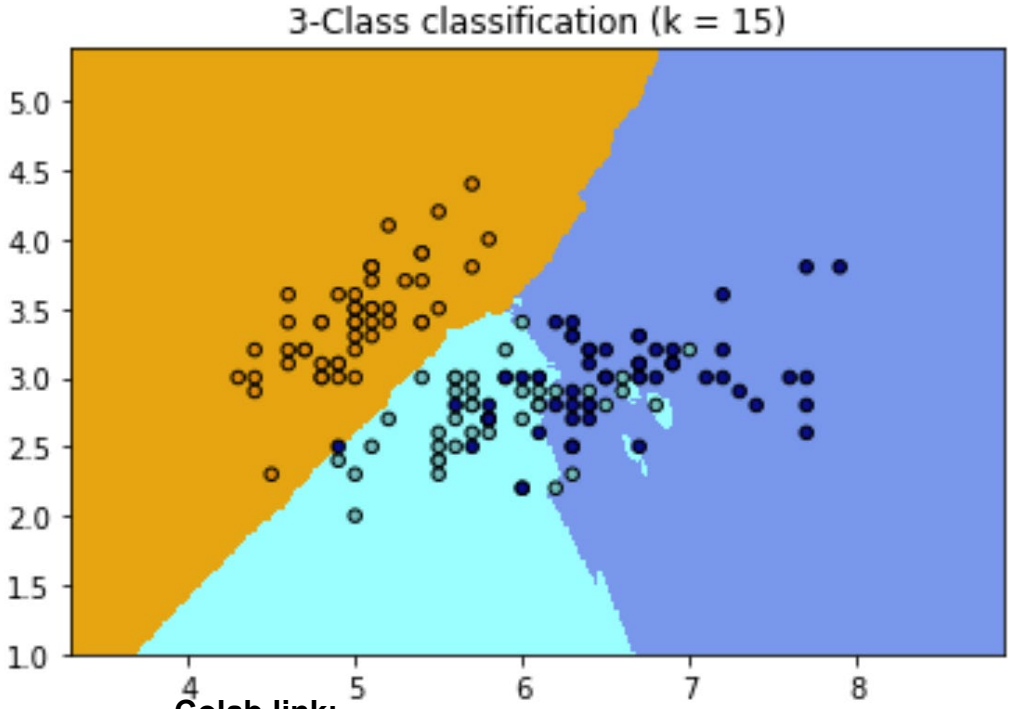
- Take-home messages:
  - Slightly outperforms k-nn (remember from our plots in slide 18, class boundaries are quite linear)
  - More sophisticated methods (SVM) do not improve, again class boundaries are quite linear
  - k-fold cross-validation IS relevant!

Iris Setosa



Iris Versicolor

Iris Virginica



Colab link:
https://drive.google.com/file/d/1GftP9OQDIe6_7dP-AhIe_QgIZGiGPOjm

Data standardization

- Many ML algorithms (SVM, L2/L1 regularization) assume all dimensions of x has zero-mean and similar covariance
- Most usual standardization:

$$x = (1, x_1, \cdots, x_d)^T \rightarrow x' = \left(1, x'_1, \cdots, x'_d\right)^T$$

$$x'_j = \frac{x_j - \mu_j}{\sigma_j}$$

- For more details, you can check https://scikit-learn.org/stable/modules/preprocessing.html
  - For example, for data with outliers, you should use robust estimates for mean and covariance

Ensembling

- The machine learning equivalent to the wisdom of the crowd
  - Aggregating predictions from several models improve over the individual ones
- Example (from Géron 2019):
  - Aggregating the votes of 1,000 binary classifiers with individual accuracies of 51% gives us an accuracy of 75%!
- **Practical trick:** when, at the end of a machine learning project, you trained several good models, combine their predictions to obtain an even better one.

Aurélien Géron, Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow, O'Reilly, 2nd edition, 2019

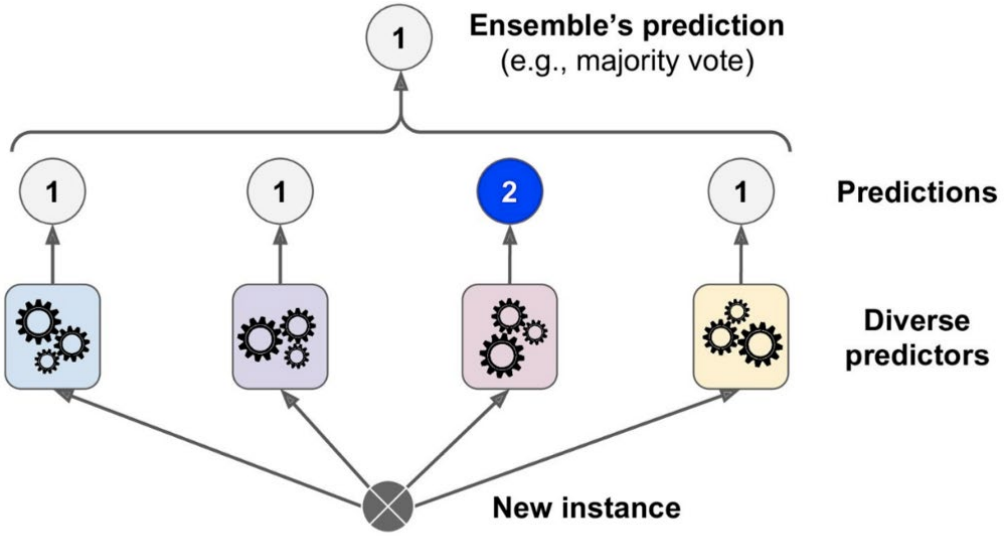
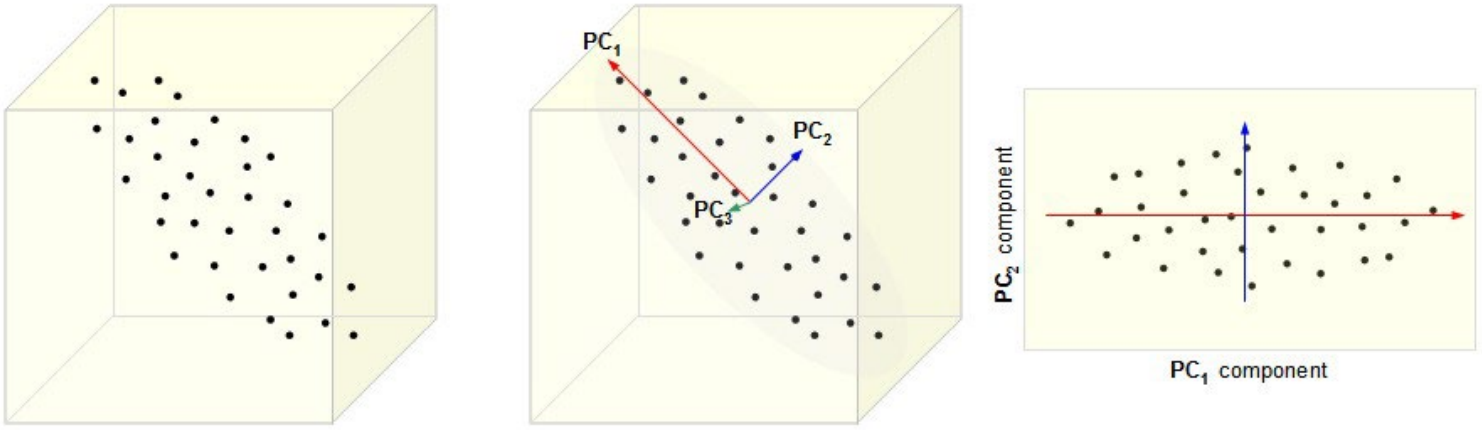


Figure 7-2. Hard voting classifier predictions

PCA (unsupervised learning)

- PCA → Principal Component Analysis.
  - **Motivation:** in high dimensional data, some dimensions are redundant, others are highly correlated…
  - **Intuition:** Project the data into a low-dimensional space preserving the dimensions of maximum variance
  - SVD decomposition ($A = U\Sigma V^T$) of the training matrix $X = [x^{(1)} \quad \dots \quad x^{(N)}]$, the columns of V are the unit vectors of the principal components.
  - The low-d data is computed as $X_{d'} = XW_{d'}$; $W_{d'} \coloneqq V[1:d']$



For more details, check Deisenroth chapter 10

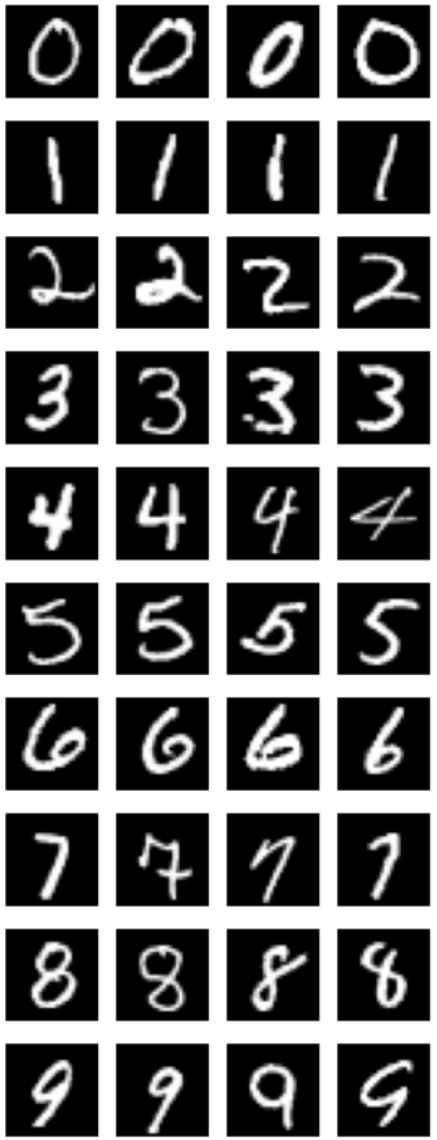
1-NN with and without PCA in MNIST

- Time per 10,000 queries in the test set
- Full-size input, $x \in \mathbb{R}^{784}$:

```
k=1, accuracy=97.22%, time=849.6 s
```

- Removing lowest-variance dimensions with PCA, up to 98% time reduction and same accuracy!

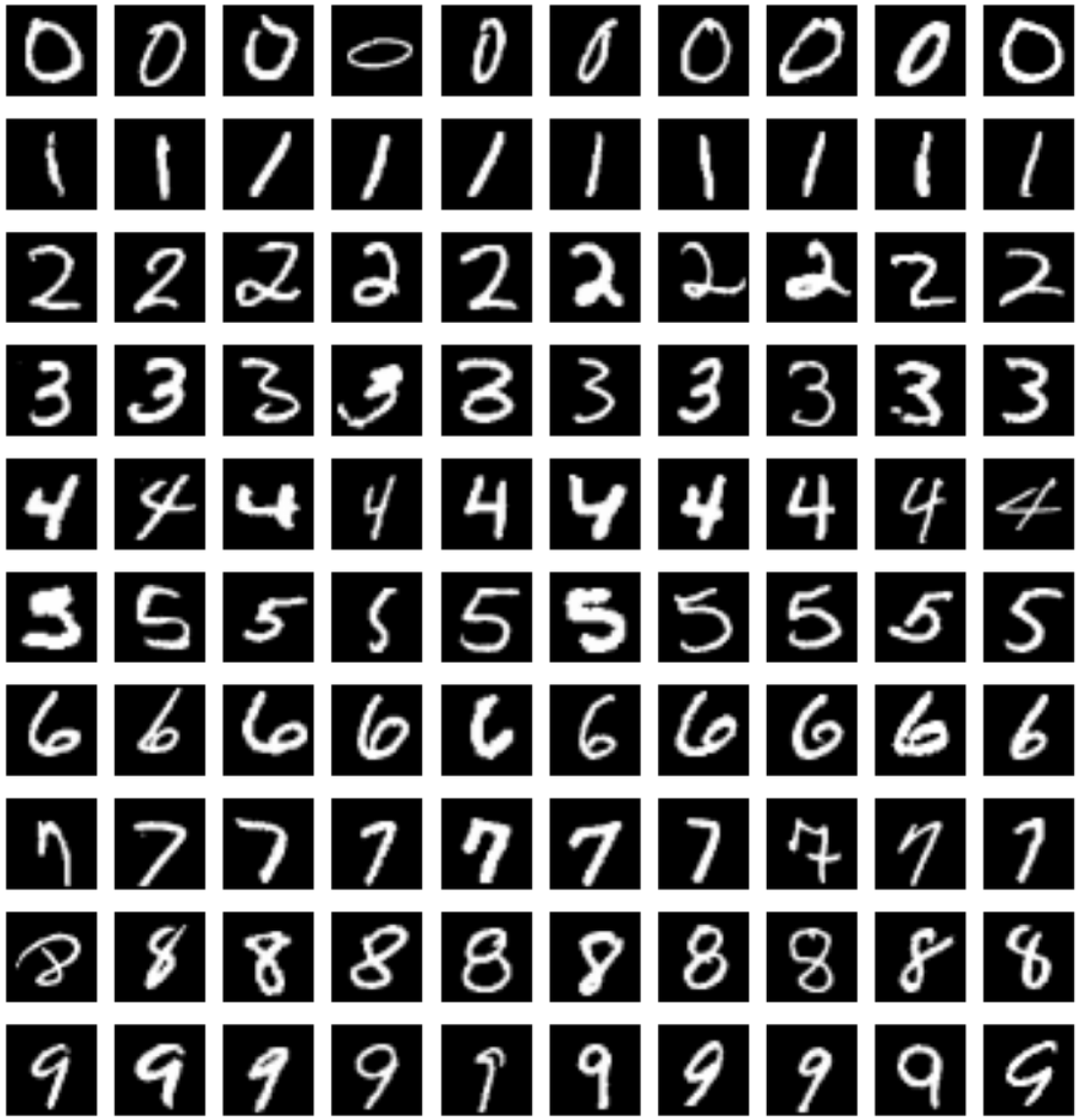
```
dims after PCA=10, accuracy=91.95%, time=1.3 s
dims after PCA=20, accuracy=96.60%, time=6.1 s
dims after PCA=30, accuracy=97.60%, time=14.8 s
dims after PCA=40, accuracy=97.43%, time=24.6 s
dims after PCA=50, accuracy=97.45%, time=36.3 s
dims after PCA=60, accuracy=97.56%, time=48.9 s
dims after PCA=70, accuracy=97.48%, time=62.0 s
dims after PCA=80, accuracy=97.51%, time=69.3 s
dims after PCA=90, accuracy=97.44%, time=81.7 s
dims after PCA=100, accuracy=97.54%, time=93.8 s
dims after PCA=200, accuracy=97.29%, time=159.8 s
dims after PCA=300, accuracy=97.24%, time=243.2 s
dims after PCA=400, accuracy=97.22%, time=318.2 s
dims after PCA=500, accuracy=97.24%, time=400.1 s
dims after PCA=600, accuracy=97.22%, time=473.3 s
```



Colab link:
https://colab.research.google.com/drive/1EdemME5vbZr01Q6CijYh0eo3hnRsjjKE

Assignment: Logistic regression in MNIST

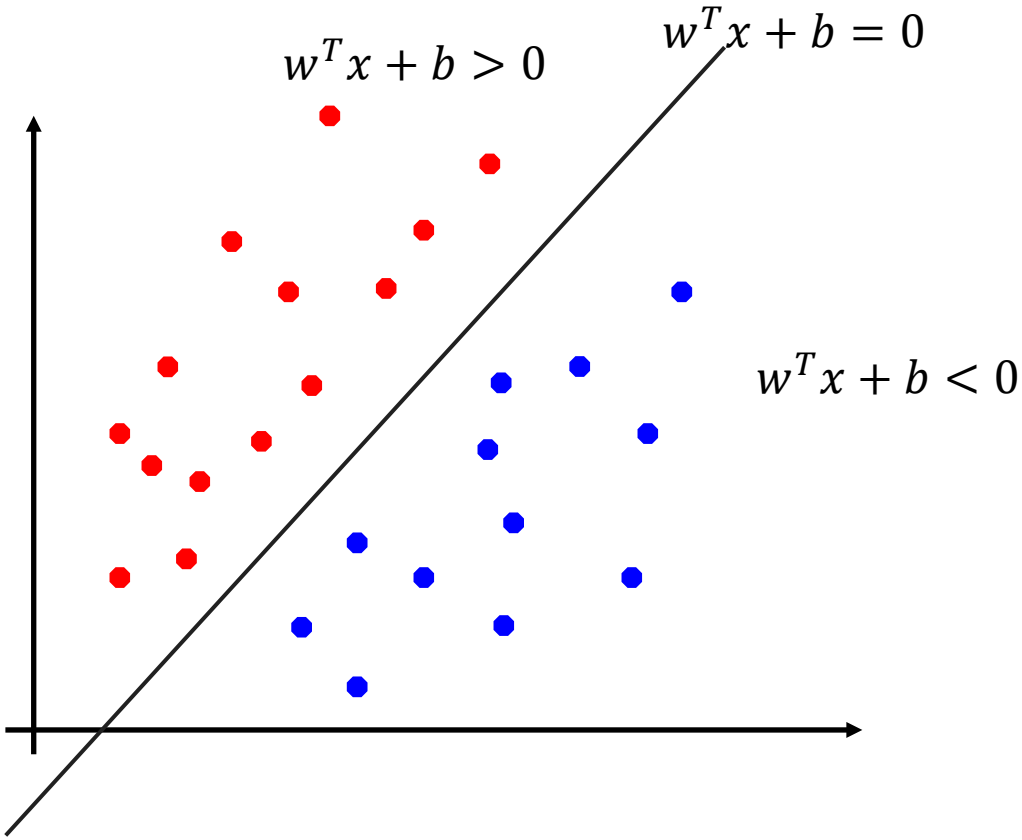
- Spoilers:
  - Excruciatingly slow!
    - Due to high dimensionality and large dataset size
    - Shortlist promising models using a reduced version of the data!
    - Use PCA!
  - 1-NN better than logistic regression…
    - why?



https://colab.research.google.com/drive/1HS9988xVWD6yqw9FRbe4z7qQfAFejL_V

Support Vector Machines (SVM)

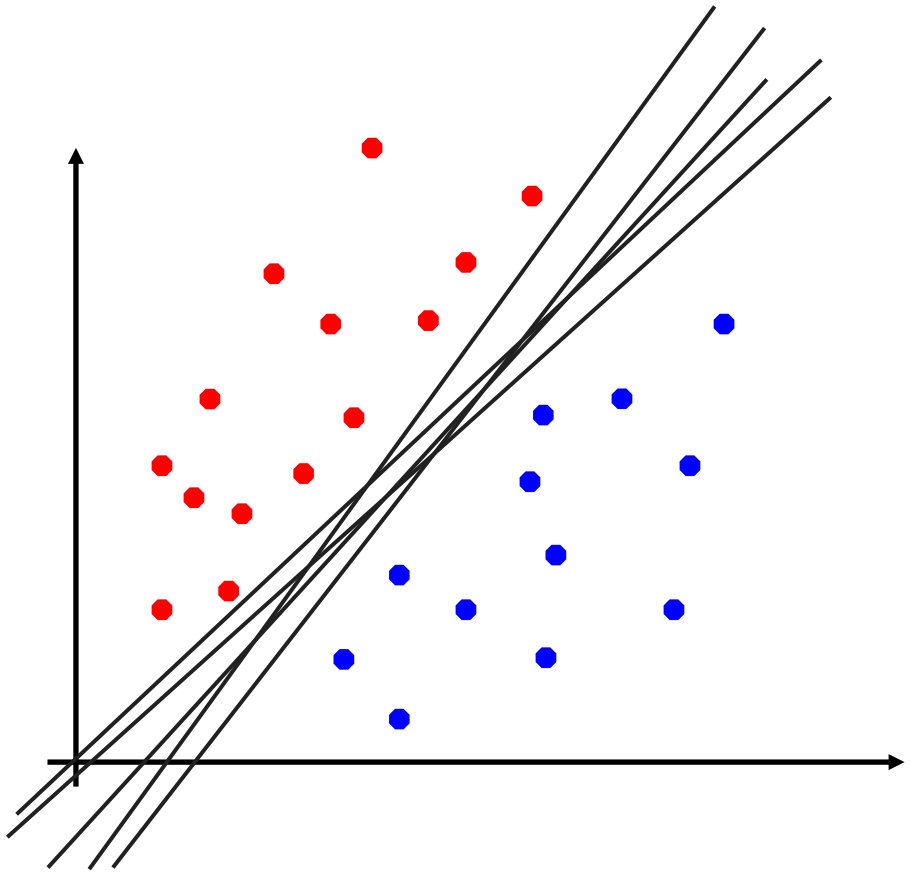
- Recap: Linear classification consists of finding an hyperplane in the feature space that separates the training data



For more details, check Deisenroth chapter 12

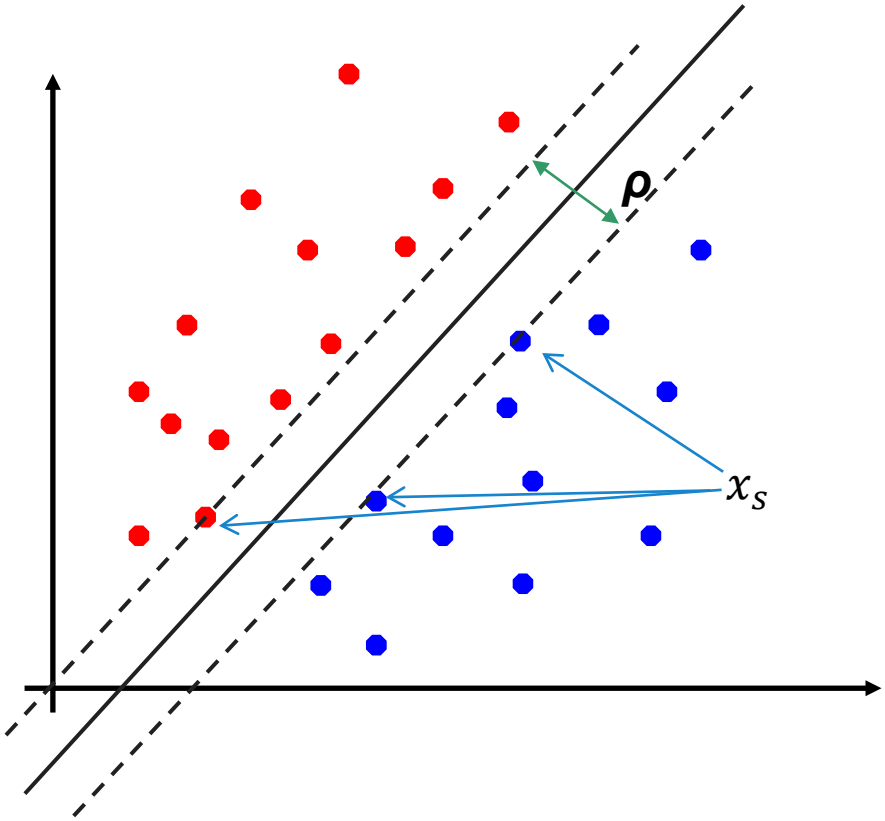
Support Vector Machines (SVM)

- But… which one of these hyperplanes would be the best one?



Support Vector Machines (SVM)

- SVM idea: let's take the one with **maximum margin** ρ
- Maximum margin:
  - The closest samples to the hyperplane are the support vectors x_s
  - The margin is the distance between the support vectors
  - Only support vectors matter! (which seems reasonable…)



"Hard-margin" SVM

- For each training example $(x^{(i)}, y^{(i)}), y^{(i)} \in \{-1, +1\}$

$$\begin{aligned} w^T x^{(i)} + b &\leq {}^{-\rho}/_{2} \quad if\ y^{(i)} = -1 \\ w^T x^{(i)} + b &\geq {}^{\rho}/_{2} \quad if\ y^{(i)} = +1 \end{aligned} \Rightarrow y^{(i)}\left(w^T x^{(i)} + b\right) \geq {}^{\rho}/_{2}$$

- If $x_s^{(i)}$ is a support vector, the above inequality is an equality
- After re-scaling w and b by ${}^{\rho}/_{2}$, the distance between each support vector $x_s^{(i)}$ and the hyperplane is ${}^{1}/_{\|w\|}$
- The margin is then $\rho = {}^{2}/_{\|w\|}$
- And the margin maximization can be formulated as

$$\hat{w}, \hat{b} = \arg max_{w,b}\ {}^{2}/_{\|w\|} \text{ (alternatively } \arg min_{w,b}\ \frac{1}{2}\|w\|^2\text{)}$$

subject to $y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1$ (for re-scaled w and b)

"Soft-margin" SVM

- "Hard-margin SVM" does not allow data to be on the wrong side of the hyperplane
- For noisy data, we might want to allow that
- Idea: introduce a per-sample slack variable $\xi^{(i)} > 0$ modelling deviations from the right side of the hyperplane
- The optimization is now

$$\hat{w}, \hat{b} = \arg min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_i \xi^{(i)}$$

subject to $y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \xi^{(i)}$ (re-scaled w and b)

- C trades off the margin and the amount of slack we have
  - $C \downarrow\downarrow$ regularizes the solution (favors simple decision functions)
  - Usually tuned by cross-validation

Non-linear SVM

- The SVM parameters are learnt by solving the dual form of the optimization

$$\hat{\alpha} = \arg max_{\alpha}\, {}^{1}/_{2} \sum_i \sum_j y^{(i)} y^{(j)} \alpha^{(i)} \alpha^{(j)} \left\langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \right\rangle - \sum_i \alpha^{(i)}$$

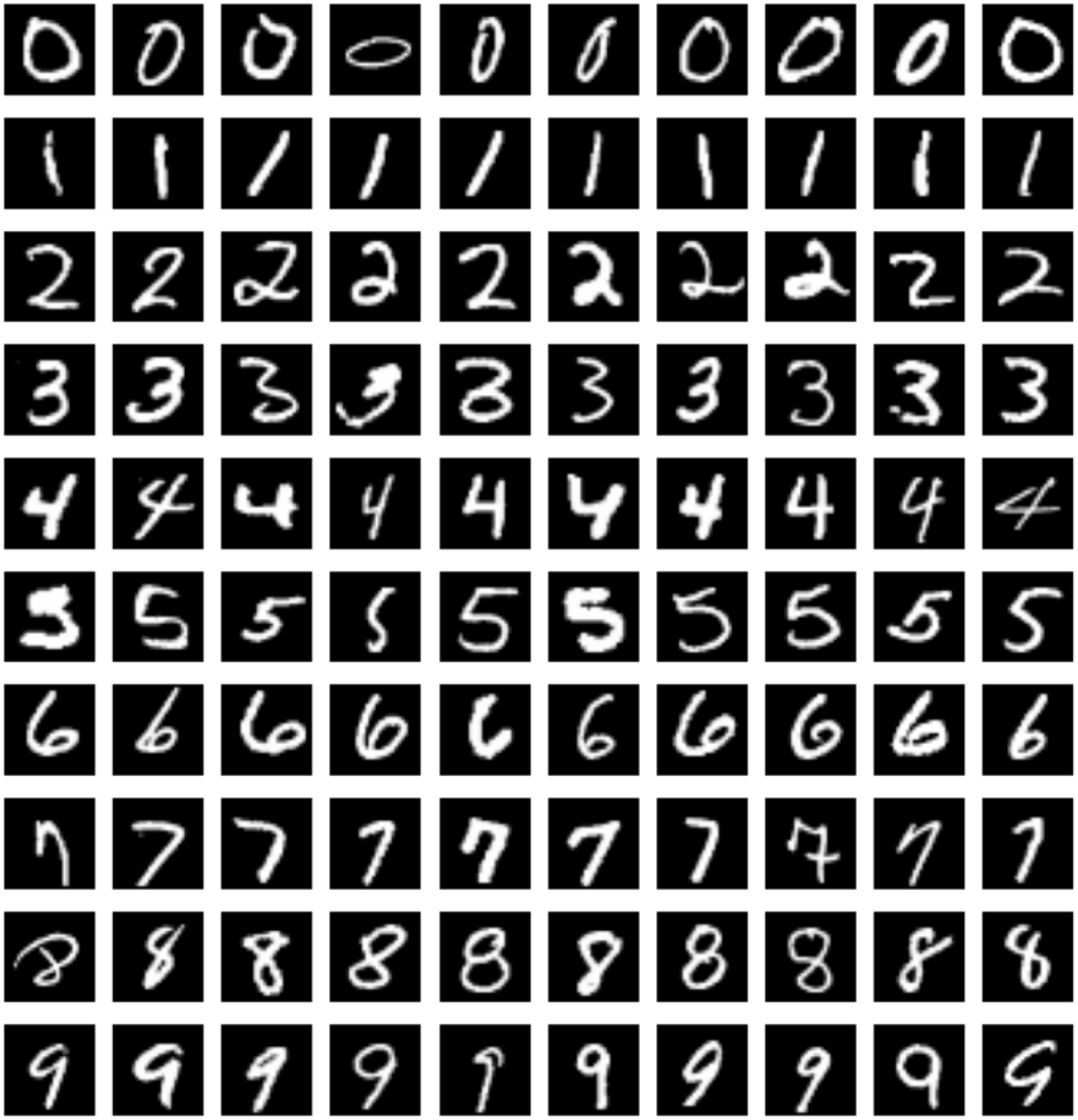
$$\text{subject to } \textstyle\sum_i y^{(i)} \alpha^{(i)} = 0,\ 0 \leq \alpha^{(i)} \leq \mathrm{C}$$

- The objective function is based in the inner product $\left\langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \right\rangle$
- For non-linear functions of the data $\phi(x^{(i)})$, the only modification is on the inner product $\left\langle \boldsymbol{\phi}(\boldsymbol{x}^{(i)}), \boldsymbol{\phi}(\boldsymbol{x}^{(j)}) \right\rangle$
- Instead of choosing the non-linear function $\phi(x^{(i)})$, we choose the similarity function $k(x^{(i)}, x^{(j)}) = \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle_{\mathcal{H}}$
- The similarity function $k(x^{(i)}, x^{(j)})$ is a kernel (more about kernels later in the course…)

Non-linear SVM hyperparameters

- Obtained by cross-validation
- C parameter
  - Trades off the margin and the slack allowed to the data
- Kernel and kernel parameters. Usual choices
  - Linear kernel: $k\left(x^{(i)}, x^{(j)}\right) = x^{(i)}x^{(j)}$
  - Polynomial kernel: $k\left(x^{(i)}, x^{(j)}\right) = \left(\gamma x^{(i)}x^{(j)} + r\right)^d$
  - Gaussian kernel: $k\left(x^{(i)}, x^{(j)}\right) = \exp\left(-\frac{\left\|x^{(i)} - x^{(j)}\right\|^2}{\sigma^2}\right)$
  - Radial basis function: $k\left(x^{(i)}, x^{(j)}\right) = \exp\left(-\gamma\left\|x^{(i)} - x^{(j)}\right\|^2\right)$
  - Sigmoid kernel: $k\left(x^{(i)}, x^{(j)}\right) = \tanh\left(\gamma x^{(i)}x^{(j)} + r\right)$

Assignment: SVM in MNIST??



https://colab.research.google.com/drive/1HS9988xVWD6yqw9FRbe4z7qQfAFejL_V

Universidad Zaragoza

Regression metrics

- **Mean absolute error** $\rightarrow \frac{1}{N}\sum_{i=1}^{N}\left|y^{(i)} - w^T x^{(i)}\right|$
  - Small influence of large errors
- **Mean squared error** $\rightarrow \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - w^T x^{(i)}\right)^2$
  - Highly influenced by large errors
- **Root mean squared error** $\rightarrow \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - w^T x^{(i)}\right)^2}$
- **Median absolute error** $\rightarrow \text{median}\left(\left|y^{(1)} - w^T x^{(1)}\right|, \ldots, \left|y^{(N)} - w^T x^{(N)}\right|\right)$
  - Robust to outliers!
- **R^2 score** $\rightarrow 1 - \frac{\sum_{i=1}^{N}\left(y^{(i)} - w^T x^{(i)}\right)^2}{\sum_{i=1}^{N}\left(y^{(i)} - \bar{y}\right)^2} \in [-\infty, 1], \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y^{(i)}$
  - Proportion of variance in y explained by the variables of the model
- Many others! Maximum error, Mean absolute percentage error, explained variance score, etc.
- Compare against dumb baseline (mean of the training predictions?)

Types of error (binary classification)

- **True Positive (TP):** Positive data (correctly) classified as positive data.
- **True Negative (TN):** Negative data (correctly) classified as negative data.
- **False Positive / Type I Error (FP):** Negative data (not correctly) classified as positive data.
- **False Negative / Type II Error (FN):** Positive data (not correctly) classified as negative data.

	Ground Truth		
Prediction		Positive	Negative
	Positive	TP	FP
	Negative	FN	TN

Metrics: Precision and Recall

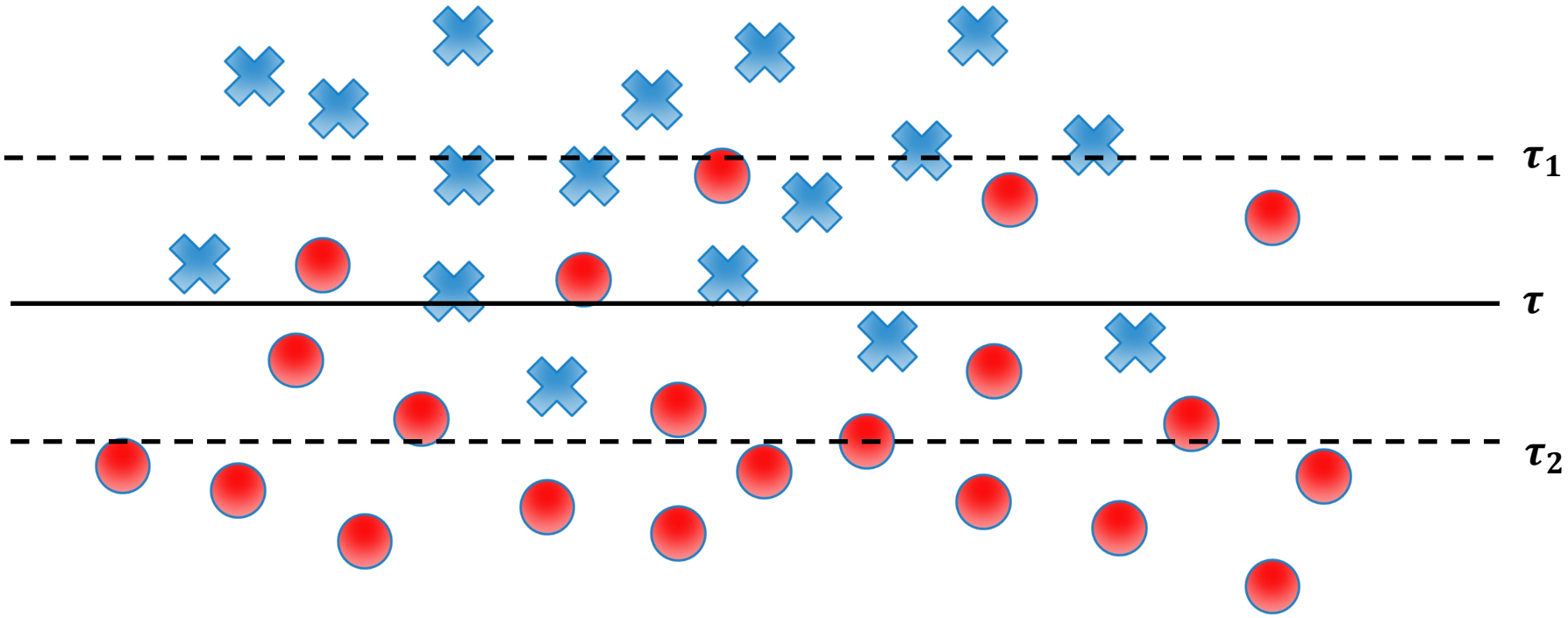
- **Accuracy, success rate, hit rate** $\rightarrow \frac{TP+TN}{TP+FP+TN+FN}$
  - Limitation: Not every error is equally important…
  - Best and worst success rate?
- **Precision** $\rightarrow \frac{TP}{TP+FP} \in [0,1]$
  - Ratio of the true positives over all the positive classifications.
  - Fraction of retrieved elements that are relevant.
- **Recall** $\rightarrow \frac{TP}{TP+FN} \in [0,1]$
  - Ratio of the true positives over all the positive samples.
  - Fraction of relevant elements that are retrieved.
- Both depend on the classification threshold.
- They are dependent, if one increases the other decreases. Sometimes given in pairs (e.g., precision at recall X).

Precision and Recall (some intuition)

- The girlfriend (boyfriend) example:
  - Your girlfriend/boyfriend makes birthday presents for you every year.
  - One year she/he asks: Do you remember all my presents from the last 10 years?
  - **Precision** is how many are correct from the ones you said you remember.
  - **Recall** is how many correct you can remember from all correct ones.
  - How can we have 100% precision or recall? What happens to the other?
  - What is the relation to the classification threshold?
- The airport security example:
  - FP and FN: Have both errors the same consequences?
  - How can we have 100% precision or recall?
- Do you need precision, recall or both?
  - The solution is, most often, a compromise…

Precision-Recall and thresholds

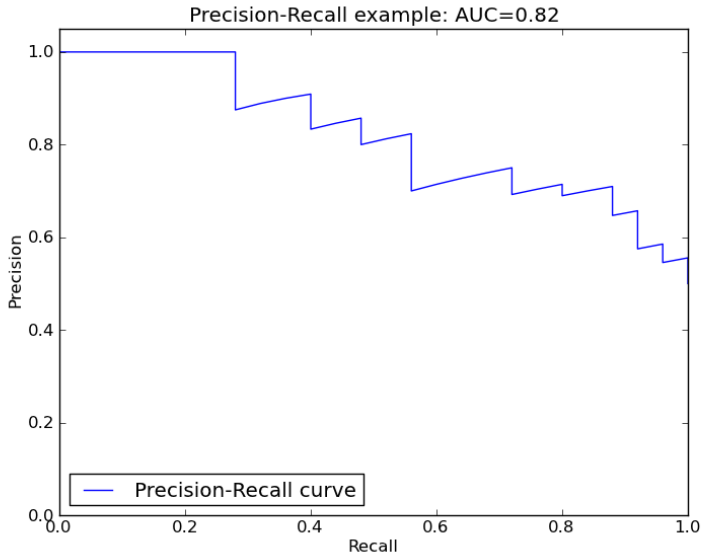
- Two-class problem



- What happens to the TP, TN, FP, FN, precision and recall if we move the boundary from τ to τ_1 or τ_2?

Precision-Recall curve

- 2D plot, recall in x-axis and precision in y-axis.
- It shows the precision-recall pairs when we change the classifier threshold τ
- $\tau \uparrow \Rightarrow FP \downarrow \Rightarrow FN \uparrow \Rightarrow P \rightarrow 1 \;\Rightarrow R \rightarrow 0$
- $\tau \downarrow \Rightarrow FP \uparrow \Rightarrow FN \downarrow \Rightarrow P \rightarrow 0 \;\Rightarrow R \rightarrow 1$
- Changing a classifier threshold τ it operates in different modes. We have to select the more appropriate for our purposes.
- For general purposes, the area under the curve (AUC) is usually taken as a general metric (the higher the better).
- What curve and AUC would have a perfect classifier? What curve and AUC would have the worst possible classifier?



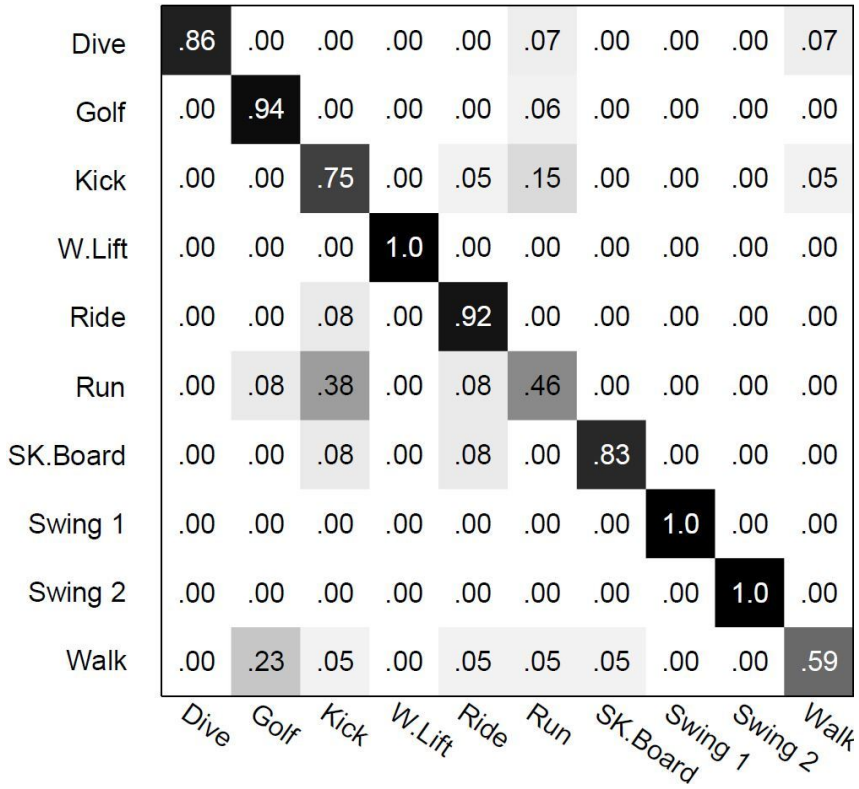
F1 score, confusion matrix

- **F1 score**
  - Harmonic mean of precision and recall (the higher the better)

$$F_1 = \frac{2}{{}^1/_P + {}^1/_R} = 2\frac{precision * recall}{precision + recall} \in [0,1]$$

  - It can be weighed to give β times more importance to precision than recall $F_1 = (1+\beta^2)\frac{precision * recall}{\beta^2 * precision + recall}$

- **Confusion matrix**
  - Rows: true categories.
  - Columns: predicted categories.
  - Cells: number/frequency of ocurrence.
  - More information than other metrics (for example, typical classification mistakes)
  - Difficult to display if many categories.
  - Difficult to summarize…

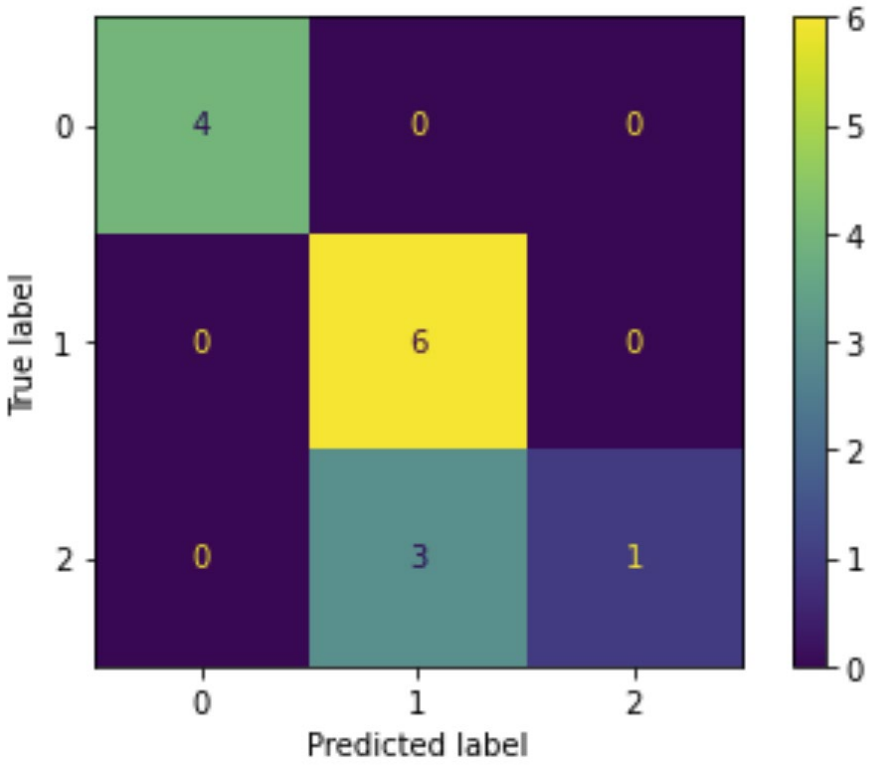


Confusion matrix for action recognition from videos

Taken from Qiu, Jiang, and Chellappa, "Sparse Dictionary-based Representation and Recognition of Human Action Attributes", ICCV 2011

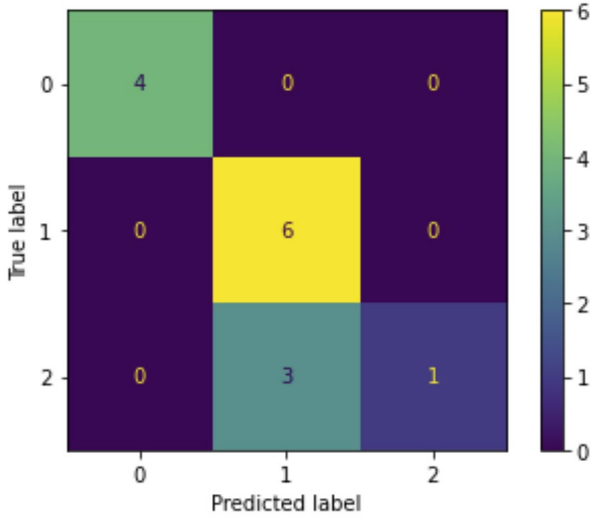
TP, TN, FP and FN from the confusion matrix

- Remember:
  - Rows: true categories.
  - Columns: predicted categories.
- **True positives per class:**
  - Diagonal elements [4, 6, 1]
- **False positives per class:**
  - Sum of the off-diagonal elements of each column [0, 3, 0]
- **False negatives per class:**
  - Sum of the off-diagonal elements of each row [0, 0, 3]
- **True negatives per class:**
  - Sum of all matrix elements except the row and column of the class [10, 5, 10]



Multiclass Precision/Recall

- **True positives per class:** [4, 6, 1], **True negatives per class:** [10, 5, 10]
- **False positives per class:** [0, 3, 0], **False negatives per class:** [0, 0, 3]



	Accuracy	Precision	Recall
Class 1	$\frac{4+10}{4+6+1+3} = \mathbf{1.0}$	$\frac{4}{4+0} = \mathbf{1.0}$	$\frac{4}{4+0} = \mathbf{1.0}$
Class 2	$\frac{6+5}{4+6+1+3} = \mathbf{0.79}$	$\frac{6}{6+3} = \mathbf{0.66}$	$\frac{6}{6+0} = \mathbf{1.0}$
Class 3	$\frac{1+10}{4+6+1+3} = \mathbf{0.79}$	$\frac{1}{1+0} = \mathbf{1.0}$	$\frac{1}{1+3} = \mathbf{0.25}$

- **Overall accuracy** $= \frac{correct}{total} = \frac{4+6+1}{4+6+1+3} = 0.78$
- **Average accuracy** $= \frac{1.0+0.79+0.79}{3} = 0.86$
- **Average Precision** $= \frac{1.0+0.66+1.0}{3} = 0.89$
- **Average Recall** $= \frac{1.0+1.0+0.25}{3} = 0.75$

If data is imbalanced, average metrics may be weighted…

Universidad Zaragoza
1542

Machine Learning (69152)

Máster in Robotics, Graphics and Computer Vision