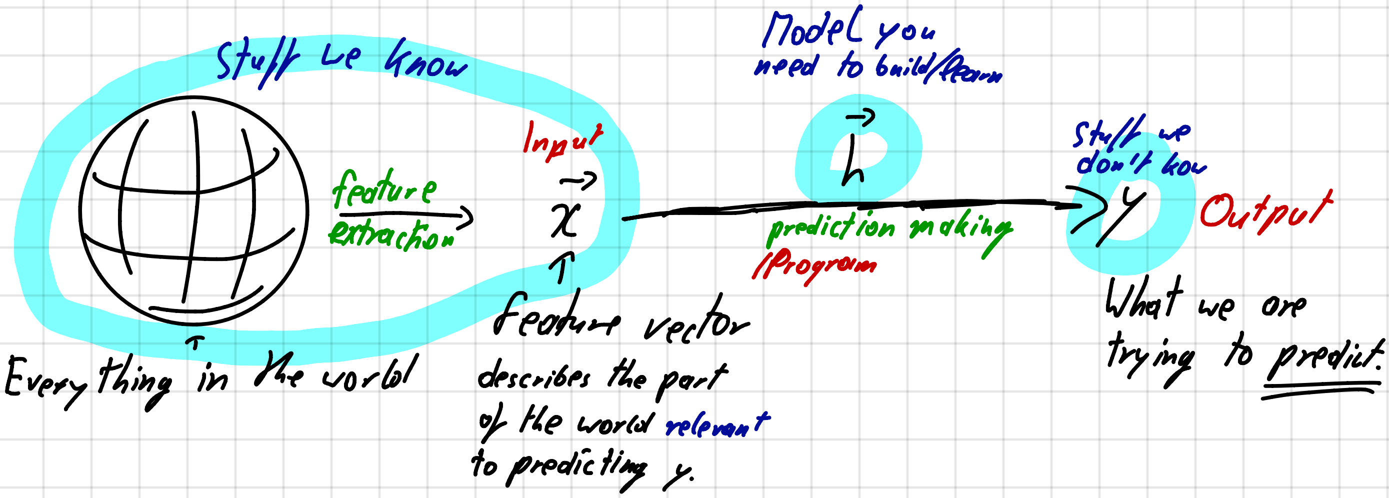
CS 4780

ML Setup

prediction making:



Feature extraction and the actual prediction making are both important components, and both incur error.

Exercise: What could $\vec{x}$ consist of when you are trying to predict...

... if an email is spam or not-spam (=ham)
... if the Coca Cola stock will go up tomorrow?
... where Jupiter will be tomorrow?
... if a picture that you just took contains a Moose?

How do we obtain h?

- Traditional CS approach: Pay CS Major to implement a program (somehow)
- Supervised Learning approach: Learn h from past data!

To learn we need four things:

1. Data $\{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\}$ for which we know both $\vec{x}$ and y
2. A hypothesis class $\mathcal{H}$ of candidate models $(h \in \mathcal{H})$
3. A loss function $\ell: \mathcal{H} \to \mathbb{R}_0^+$ that tells us how good $h \in \mathcal{H}$ is.
4. An optimization algorithm to find $h^* = \underset{h \in \mathcal{H}}{\arg\min}\, \ell(h)$

Intro to Machine Learning

Supervised Learning

Learn to make predictions from data.

Setup: Data $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_n, y_n)\} \subseteq \mathbb{R}^d \times C$

$\mathbb{R}^d$: d-dimensional feature space
$\vec{x}_i$: input vector of i-th input sample
y_i : label of i-th sample
C : label space

Multiple scenarios for C:

- **binary classification** $|C| = 2$ $\quad C = \{0,1\}$ or $C = \{-1, +1\}$

  example: Spam filtering. An email is either spam (+1) or not spam (−1)

- **multi-class classification** $|C| > 2$: $C = \{1, \dots, K\}$

  example: face classification. A person can be exactly one of K identities (e.g. 1 = "Barack Obama", 2 = "George W. Bush")

- **Regression** $C = \mathbb{R}$ $\quad C = \mathbb{R}$

  example: predict temperature or height of a person

The goal of supervised learning is to find a function $h: \mathbb{R}^d \to C$
such that $h(\vec{x}_i) = y_i$ for $(\vec{x}_i, y_i) \in D$ (training)
and $h(\vec{x}_t) = y_t$ for $(\vec{x}_t, y_t) \notin D$ (testing)

We call $\vec{x}_i$ a feature vector and the d dimensions the <u>features</u> describing the i-th sample.

<u>Examples of feature vectors:</u>

- <u>Patient data</u> in a hospital ($d \leq 100$, dense)

$$\vec{x}_i = \begin{pmatrix} 1 \\ 183 \\ 67 \\ \vdots \end{pmatrix} \begin{array}{l} \leftarrow \text{male/female} \\ \leftarrow \text{height in cm} \\ \leftarrow \text{age in years} \\ \end{array}$$

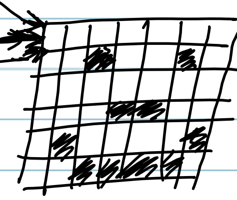
- <u>Text document</u> in bag-of-words format: ($d \approx 100\,000 - 10M$ sparse)

$$\vec{x}_i = \begin{pmatrix} 10 \\ 0 \\ \vdots \\ 2 \\ \vdots \\ 0 \end{pmatrix} \begin{array}{l} \leftarrow \text{occurances of word "a"} \\ \leftarrow \text{— " — "ant"} \\ \\ \leftarrow \text{— " — "like"} \\ \\ \leftarrow \text{— " — "zynga"} \end{array}$$

(we call a feature vector <u>sparse</u> if it consists of mostly zeros.)

- <u>Image</u>: ($d \approx 300.000 - 10M$) (dense)

$$x_i = \begin{pmatrix} 0.1 \\ 0.7 \\ 0.9 \\ \end{pmatrix} \begin{array}{l} \leftarrow \text{red value} \\ \leftarrow \text{green value} \\ \leftarrow \text{blue value} \\ \end{array}$$



A 7MP camera pic results in $7 \times 3 = 21$ million features.

Loss functions (aka risk function)

We want to find the best $h()$ for a data set D.
How do we quantify "best"?
There are many choices. Famous examples:

- 0/1 loss: $\frac{1}{n}\sum_{i=1}^{n} \underbrace{\delta_{h(\vec{x}_i)\neq y_i}}_{\begin{cases}1 & \text{if } h(\vec{x}_i)\neq y_i \\ 0 & \text{o/w}\end{cases}}$ ← returns error rate on

- squared loss $\frac{1}{n}\sum_{i=1}^{n} \left(h(\vec{x}_i)-y_i\right)^2$
- absolute loss $\frac{1}{n}\sum_{i=1}^{n} \left|h(\vec{x}_i)-y_i\right|$

} Typically used for regression

Generalization

If you find a function $h()$ with low loss on your data D, how do you know it will still get examples right that are not in D?

Bad example "memorizer" $h()$:

$$h(x) = \begin{cases} y_i & \text{if } \exists (\vec{x}_i, y_i) \in D \text{ s.t. } \vec{x}_i = \vec{x} \\ 0 & \text{o/w} \end{cases}$$

Gets perfect 0% error on training data D, but does horribly with samples not in D.

Train/test splits:

Split your data into train, validation and test splits.

	D_{TR}	D_{VA}	D_{TE}
D	TRAIN	VALIDATION	TEST
Example:	80%	10%	10%

QUIZ: Why do we need D_{VA}?

Choose $h()$ on TRAIN (D_{TR}) and evaluate on D_{TE}.

How should you split the data?

- <u>By time</u> if there is a temporal component
- <u>uniformly at random</u> if data is i.i.d.

The test error (or test loss) approximates the true generalization error/loss.

<u>Let's put it all together:</u>

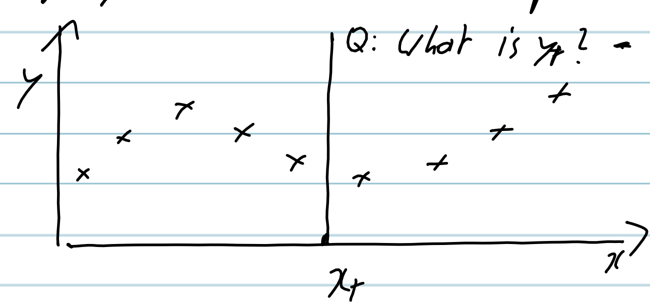
<u>Learning</u>	<u>Evaluation</u>	<u>Generalization</u>
$\underset{\text{hypothesis class (set of all possible classifiers } h)}{\underset{h \in \mathcal{H}}{h = \operatorname{argmin}}} \underbrace{\frac{1}{\lvert D_{TR} \rvert} \sum_{(\vec{x}, y) \in D_{TR}} \ell(\vec{x}, y)}_{\text{training loss}}$	$\underbrace{\mathcal{E}_{TE} = \frac{1}{\lvert D_{TE} \rvert} \sum_{(\vec{x}, y) \in D_{TE}} \ell(\vec{x}, y)}_{\text{testing loss}}$	$\underbrace{\mathcal{E} = E\left[\ell(\vec{x}, y)\right]_{(\vec{x}, y) \sim P}}_{\text{generalization loss (unobtainable)}}$

Quiz: Why does $\mathcal{E}_{TE} \xrightarrow{\lvert D_{TE} \rvert \to \infty} \mathcal{E}$?

<u>No Free lunch</u> (Every ML algorithm makes assumptions. Always know which!)

Which hypothesis class $\mathcal{H}$ should you choose?
It depends on the data. $\mathcal{H}$ encodes <u>your assumptions</u> about the data set / distribution P.
There is no one perfect $\mathcal{H}$ for all problems (no free lunch theorem)

<u>Example:</u>



Q: What is y_t? - A: <u>Impossible to know without assumptions.</u>