know where to look for

- http://neuralnetworksanddeeplearning.com/ Chapter 1
- d2l.ai Chapter 3 and Chapter 4
- https://numpy.org/doc/

theory:

- the four basic components of ML problems
  - Input and Output space examples
  - The prediction model
  - Loss function - how good is the prediction of the model ?
  - the set/space of all possible prediction models

more theory:

- classification with the affine model
- the inner product and its properties

Outline

Section on: **What are the essential components in machine learning problems?**

Next: Lets consider a very simple prediction problem.

We want to predict the selling price of a species of aquarium fish - depending on its size and its color intensity.

- The intended output is: y - selling price in EUR
- we predict it based on two features: x_0 - size in cm, x_1 - color intensity, a dimensionless measurement from the interval $[0, 1]$
- our prediction model has to map a vector of two measurements (x_0, x_1) onto a price y

- we have as inputs: the set of all 2 dimensional vectors (features column)

$$\mathbb{R}^2 = \{(x_0, x_1), x_i \in \mathbb{R} \text{ a real number}\}$$

- goal: we want to output for any x a real value y
- a prediction model which can do that:

$$f(x) = w_0 x_0 + w_1 x_1 + b = \sum_{i=0}^{1} w_i x_i + b$$

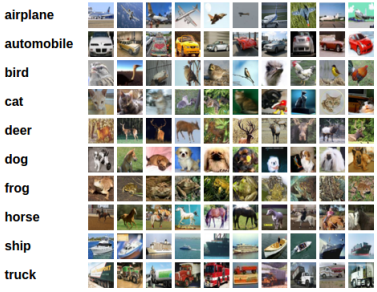
  - simple mechanism: each feature dimension x_i is weighted with w_i, then summed.
  - open question: how to find w_0, w_1, b so that the model will output sale prices close to those observed in the real market?

A Regression example:

We can see three components in this example:

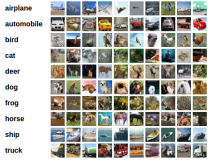
- an input space $\mathcal{X} = \mathbb{R}^2 = \{(x_0, x_1)\}$
- an output space $\mathcal{Y} = \mathbb{R}$
- a prediction mapping $f : \mathcal{X} \to \mathcal{Y}$

$\longrightarrow$ Lets look at more examples.



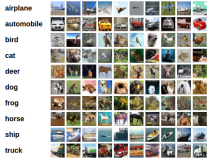
credit: thankfully taken from https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

- input: an image
- output:
  - binary classification: prediction labels usually $\{-1, +1\}$ or $\{0, 1\}$
  - **multi-class classification:** mutually exclusive classes
  - **multi-label classification:** classes can appear together in the same image/input sample



credit: thankfully taken from https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

- output in math:
  - binary classification: prediction labels usually $\{-1, +1\}$ or $\{0, 1\}$
  - **multi-class classification:** class-labels can be represented as C numbers: $y \in \{0, \ldots, C-1\}$
  - **multi-label classification:** a vector $z = (z_0, \ldots, z_{C-1})$, such that $z_t \in \{0, 1\}$
    classes can appear together, like C independent binary classification problems



credit: thankfully taken from https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

- output in code:
  - binary classification: a variable which takes 2 values
  - multi-class classification:
    - a variable which takes C values,
    - or a one-hot vector in C dimensions $(0, \ldots, 1, \ldots, 0)$
  - multi-label classification: a vector with C dimensions with two values in every dimension (e.g. 0 and 1) – sum of the zero vector and different one-hot vectors

What is the input? What is the space of all images?

- an RGB image has 3 channels, a height dimension H, a width dimension W
- possible code representations:
- a vector (1-array) with $3HW$ dimensions
- a matrix (2-array) with $(3, HW)$ dimensions or $(3W, H)$
- a 3-array with shape $(3, H, W)$
- common in deep learning: processing a set ('batch') of images, so usually a 4-array or shape $(b, 3, h, w)$ where b is the number of images in a batch
- RGBA, multi-channel hyper spectral images (satellites)

- a n-array is an array with a n-dimensional shape vector.
- in pytorch and other deep learning applications they are called n-tensors

```
a=np.zeros((3,5,2)) # 3-array
b= torch.zeros((3,5,2)) # the same in pytorch
c=np.zeros((16,3,256,256)) # 4-array
d= torch.zeros((16,3,256,256)) # the same in pytorch
```

- the mathematical definition of a tensor is something very different

what is the input? Space of images? Mathematically?

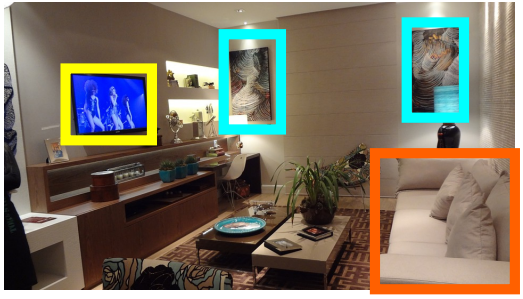
- For RGB (3 channels) , and LDR values and all possible heights and widths:

$$\cup_{h\geq 1, w\geq 1} Z^{3\times h\times w}$$
$$Z = \{0, \ldots, 255\}$$

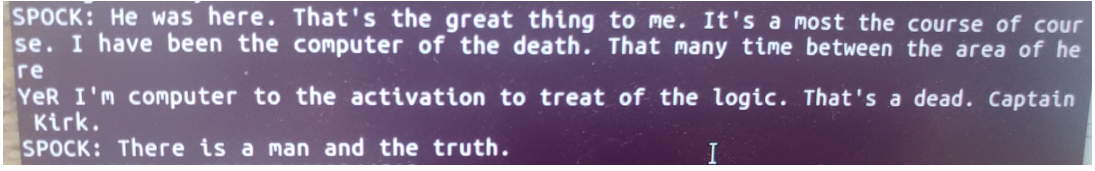
- for HDR Z is the space of all 32-bit floating point numbers.
- in many toolboxes images are normalized so that every pixel lies in the interval $[0, 1]$, then $Z = [0, 1]$
- message: there are many spaces to represent an image. If height and width are variable, it is a union of spaces with different dimensions

- Input: image.
- Output: a number of bounding boxes
- How can one describe “a number of bounding boxes” in code ?

- Input: image.
- Output: a number of bounding boxes
- How can one describe "a number of bounding boxes" in code ?
  - e.g. a list of vectors. Each vector has 5 dimensions: (x_l, y_l, x_r, y_r, c)
  - (x_l, y_l) – upper left box coordinates
  - (x_l, y_l) – lower right box coordinates
  - c – class label for the box



- Input: image.
- Output: a number of bounding boxes
- How can one mathematically describe "a number of bounding boxes" ?
  - (x_l, y_l) – upper left box coordinates
  - (x_l, y_l) – lower right box coordinates
  - c – class label for the box



- Input: sequence of words $(w_1, \ldots, w_K)$
- Output: sequence of words $(w_1, \ldots, w_L)$
  (Example: "The text is about Star Trek. It mentions Spock and James T. Kirk.")
- Input: sequence of words $(w_1, \ldots, w_K)$
- Output: set of words $\{w_1, \ldots, w_L\}$ – setup can be a sequence of classification outputs, often sequential outputs from a RNN
  (Example: "Star Trek, RNN-generated nonsense")

D3: Battle D4: Battle 2

- Input: sequence of states (health, ammo, images of view)^K
- Output: next action (move left, fire, ...) – C discrete states as in classification, but the machine learning problem is a different one
- Dosovitsky et al, ICLR 2017, https://arxiv.org/pdf/1611.01779.pdf

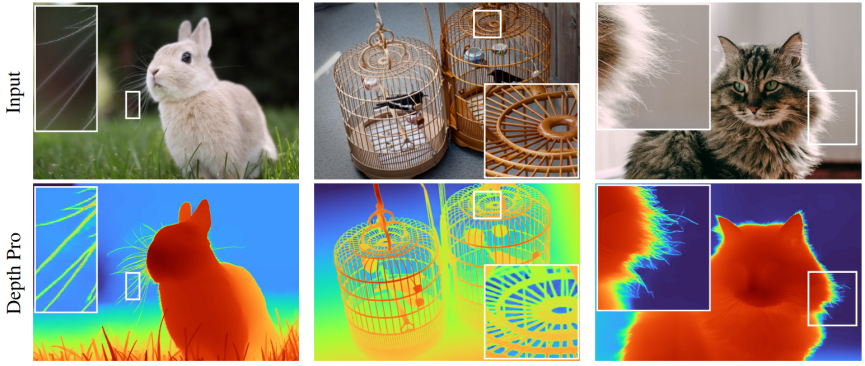
credit: https://okvqa.allenai.org/

- Input: Image + sequence of words
- Output: set of words – setup can be a sequence of classification outputs

credit: DeepLabv3 https://arxiv.org/pdf/1706.05587

- Input: Image
- Output: Segmentation label per pixel [1]

[1] this is semantic segmentation, not instance segmentation



credit: DepthPro https://arxiv.org/pdf/2410.02073

- Input: Image
- Output: log-depth per pixel $\sim$ distance to camera per pixel

Takeaway:

- input and output spaces can have a non-trivial structure

next: what do we need to describe any machine learning problem?

the components of a machine learning problem

- input space and output space
  - need to be able to represent them in code, and to describe it mathematically
- The prediction model f which we can use to predict on samples from the input space.
- A loss function $L(\cdots)$
  - it measures the quality of our predictions $f(x)$ made by the model f on samples x
- if we also train models, then we need additionally:
  - a description of the class of all possible prediction models
  - a way to select one model from this class based on a training dataset (example above: .fit(...))

What else does one need for defining a supervised machine learning problem?

So far we had:

- input spaces
- output spaces

Next:

- loss functions

- We have a pair (x, y) of input sample x and ground truth label y
  - regression model $x = (x_0, x_1)$ a 2-dim vector, $y \in \mathbb{R}^1$ a real number (fish sale price)
  - classification example above
  - bounding box example above

A loss function L:

The purpose of loss functions

We have a sample (x, y). The loss function L describes how good the prediction $f(x)$ of the model f on the input sample x is compared to the ground truth label y of the sample.

- a quality measure for your prediction by comparing the prediction $f(x)$ on x to the ground truth y of the pair (x, y)

A loss function L:

The purpose of loss functions

We have a sample (x, y). The loss function L describes how good the prediction $f(x)$ of the model f on the input sample x is compared to the ground truth label y of the sample.

- It usually takes as input a single pair (x, y) or a set of samples with respective ground truths $Z = \{(x^{\{1\}}, y^{\{1\}}), (x^{\{2\}}, y^{\{2\}}), \ldots, (x^{\{n\}}, y^{\{n\}})\}$
- It returns usually a real number
  - $L(f(x), y)$ for the single pair or
  - $L(\{(f(x^{\{1\}}), y^{\{1\}}), (f(x^{\{2\}}), y^{\{2\}}), \ldots, (f(x^{\{n\}}), y^{\{n\}})\})$ for the set of samples, respectively
  - Relative meaning: Lower loss is better.

A typical loss for regression is the root mean square error

$$\widehat{L}((x^{\{1\}}, y^{\{1\}}), \ldots, (x^{\{n\}}, y^{\{n\}})) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(x^{\{i\}}) - y^{\{i\}})^2}$$

- measure the squared deviation $(a-b)^2$ between prediction $f(x^{\{i\}})$ and ground truth $y^{\{i\}}$
- then computes the average over the data set of it
- then takes the square-root

why used ?

- Zero for a perfect match. Symmetric error around the truth y, Simple to compute, easy to understand, gradient everywhere defined

Another typical loss for regression is the mean average error

$$\widehat{L}((x^{\{1\}}, y^{\{1\}}), \ldots, (x^{\{n\}}, y^{\{n\}})) = \frac{1}{n} \sum_{i=1}^{n} |f(x^{\{i\}}) - y^{\{i\}}|$$

- measures the absolute deviation between prediction and ground truth
- then computes the average over the data set of it
- why used ? Symmetric error around the truth y, Simple to compute, easy to understand,
- the error for a single sample would be the absolute deviation $|f(x^{\{i\}}) - y^{\{i\}}|$

How are the RMSE and MAE related to each other?

RMSE and MAE belong to the same class of loss functions:

- take for one sample (x_i, y_i) the function $z_i = |f(x_i) - y_i|$
- define the generalized mean as for $z_i \geq 0$ and $p > 0$ as:

$$m_p(z_1, \ldots, z_n) = \left(\frac{1}{n} \sum_{i=1}^{n} z_i^p \right)^{1/p}$$

- note:

$$m_1(z_1, \ldots, z_n) = \frac{1}{n} \sum_{i=1}^{n} z_i \qquad \text{arithmetic mean}$$

$$m_2(z_1, \ldots, z_n) = \frac{1}{n} \sqrt{\sum_{i=1}^{n} z_i^2} \qquad \text{quadratic mean}$$

$$\lim_{p \to 0} m_p = \prod_{i=1}^{n} z_i^{1/n} \qquad \text{geometric mean}$$

$$\lim_{p \to \infty} m_p = \max(z_1, \ldots, z_n) \qquad \text{the maximum}$$

RMSE and MAE belong to the same class of loss functions:

- take for one sample (x_i, y_i) the function $z_i = |f(x_i) - y_i|$
- define the generalized mean as for $z_i \geq 0$ and $p > 0$ as:

$$m_p(z_1, \ldots, z_n) = \left(\frac{1}{n} \sum_{i=1}^{n} z_i^p \right)^{1/p}$$

- then $RMSE = M_2(z_1, \ldots, z_n)$, $MAE = M_1(z_1, \ldots, z_n)$
- $p < q$ then $m_p(z_1, \ldots, z_n) \leq m_q(z_1, \ldots, z_n)$
  meaning of this: for larger p, the mean is more sensitive to large outliers.

RMSE is more sensitive to large outliers than the MAE.

For $z_i > 0$ one can extend generalized means to negative $p < 0$, notably:

$$z_i > 0 \Rightarrow m_{-1}(z_1, \ldots, z_n) = \frac{1}{n} \frac{1}{\sum_{i=1}^{n} \frac{1}{z_i}} \text{ harmonic mean}$$

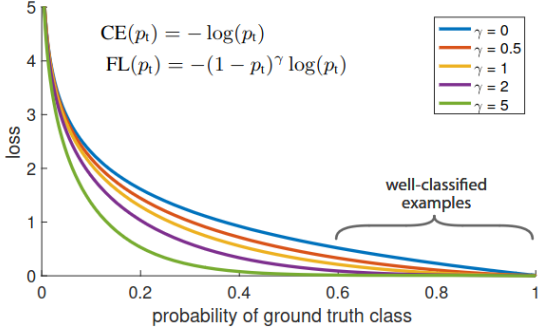
A typical loss for classification: the 0-1 error

- for a single example: produce 1 if prediction not the same as the label: $1[f(x) \neq y]$
- for a set of examples: take the average:

$$\widehat{L}((x^{\{1\}}, y^{\{1\}}), \ldots, (x^{\{n\}}, y^{\{n\}})) = \frac{1}{n} \sum_{i=1}^{n} 1[f(x^{\{i\}}) \neq y^{\{i\}}]$$

- drawback of the 0-1-error: no meaningful gradient for training (0 or undefined).

Loss functions can be non-trivial. Example Cross-entropy vs Focal loss.
https://arxiv.org/pdf/1708.02002.pdf



Focal loss seems to improve training results, because it decreases the loss for correct predictions.

Loss function roles:

Losses have 2 roles in machine learning:

- during training: find a good predictor on training data
  - for gradient based optimization the loss function needs to be differentiable (not needed with other solution strategies, e.g. genetic algorithms)
- after training: measure the quality of a predictor f on validation / test data
  - used to compare predictors
  - used to find samples with high losses (fail cases) for the next iteration of ML model development

What else does one need for defining a supervised machine learning problem?

So far we had:

- input spaces
- output spaces
- loss functions

Next:

- only briefly: classes of prediction models
- latex lectures: how to train a model.

Often it is a model f_w defined by parameters w. Then the class $\mathcal{F}$ of prediction models is the set of all models for all "allowed" parameters w

$$\mathcal{F} = \{f_w | w \in W\}$$

In our regression example:

$$f_w(x) = w_0 x_0 + w_1 x_1 + b = \sum_{i=0}^{1} w_i x_i + b$$
$$\mathcal{F} = \{f_w | w = (b, w_0, w_1) \in \mathbb{R}^3\}$$

What else does one need for defining a supervised machine learning problem?

So far we had:

- input spaces
- output spaces
- loss functions
- briefly: classes of prediction models
  skipped: how to train a model.
- thinking about input/output spaces, loss functions, classes of prediction models can help to define/identify a ML problem

Classification:

- some input space $\mathcal{X}$
- task: assign to every sample $x \in \mathcal{X}$ a class label y (Klassenzugehörigkeit/Klassenmerkmal)

Goal of classification: For every input vector $x \in \mathbb{R}^d$, correctly predict which class y it belongs.

- For 2-class classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

- we have the spaces $\mathcal{X}, \mathcal{Y}$
- next step: choose a prediction model $f : \mathcal{X} \to \mathcal{Y}$, $f(x) = y \in \mathcal{Y}$

The simplest model:

- apply a linear mapping
- predict/classify according to the sign of the output:

$$f(x) = w \cdot x + b$$

$$w \cdot x = \sum_{k=0}^{d-1} w_d x_d \text{ inner product of } w \text{ and } x$$

$$s(x) = sign(f(x)) \in \{-1, +1\}$$

$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

has the following properties:

- maps two real vectors u, v onto a real number $u \cdot v$
- linear in the first argument (u)

$$a \in \mathbb{R},\ (au) \cdot v = a(u \cdot v)$$
$$(u^{\{1\}} + u^{\{2\}}) \cdot v = u^{\{1\}} \cdot v + u^{\{2\}} \cdot v$$

  or in short:

$$(a_1 u^{\{1\}} + a_2 u^{\{2\}}) \cdot v = a_1(u^{\{1\}} \cdot v) + a_2(u^{\{2\}} \cdot v)$$

- linear in the second argument (v)

$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

has the following properties:

$$\begin{aligned} \text{Symmetry: } & u \cdot v = v \cdot u \\ v \neq \mathbf{0} \ & \Rightarrow v \cdot v > 0 \\ & \mathbf{0} \cdot u = 0 \end{aligned}$$

where $\mathbf{0}$ is the Null vector. For example in $\mathbb{R}^3$ this is the element $(0, 0, 0)$

$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

has the following properties:

- it defines a norm $\|v\|$ is a norm, that is a notion of the length of a vector v:

$$v \cdot v = \|v\|^2$$

$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

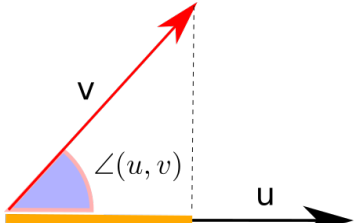
has the following properties:

- it defines an angle between two vectors:

$$\frac{u \cdot v}{(u \cdot u)^{1/2}(v \cdot v)^{1/2}} = \cos(\angle(u, v))$$

  - the angle can be measured in any dimensions
  - in higher dimensions, the angle is measured in the 2-dim plane spanned by u, v:

$$L(u, v) = \{a_0 u + a_1 v, a_0 \in \mathbb{R}, a_1 \in \mathbb{R}\}$$

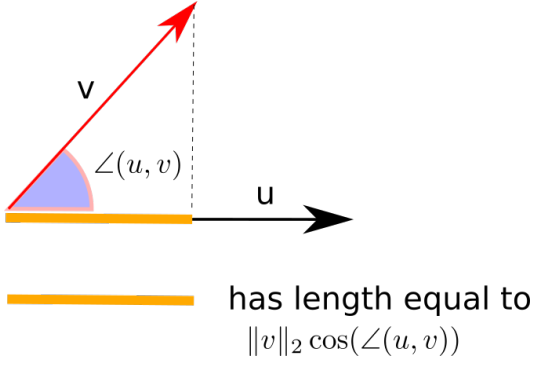


$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

- $u \cdot v$ defines an angle between two vectors:

$$\frac{u \cdot v}{(u \cdot u)^{1/2}(v \cdot v)^{1/2}} = \cos(\angle(u, v))$$

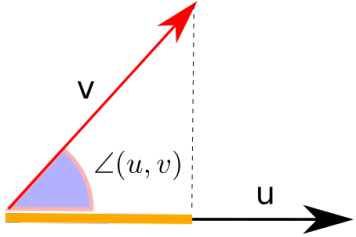
- for two vectors u, v of unit length ($\|u\|_2 = 1$) the inner product
  - lies in $[-1, +1]$
  - $u \cdot v = 1$ if $u = v$
  - gets close to 1 if their angle is close to zero,
  - gets close to 0 if their angle is close to $\pi/2 \sim 90$ deg,
  - $u \cdot v = -1$ if $u = -v$



$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

- $u \cdot v$ defines an angle between two vectors:

$$\frac{u \cdot v}{(u \cdot u)^{1/2}(v \cdot v)^{1/2}} = \cos(\angle(u, v))$$



has length equal to
$\|v\|_2 \cos(\angle(u, v))$

Interpretation of the inner product

for two vectors u, v of unit length ($\|u\|_2 = 1$) the inner product computes a similarity measure between u and v based on their angle!

Interpretation of the inner product

for two vectors u, v of unit length ($\|u\|_2 = 1$) the inner product computes a similarity measure between u and v based on their angle!

next step:

- now we can use $u \cdot v$ to define a simple classifier:

$$f(x) = w \cdot x + b$$
$$s(x) = sign(f(x)) \in \{-1, +1\}$$

Mechanism:

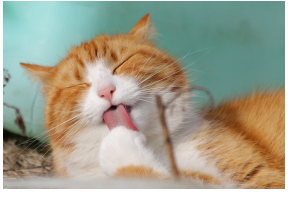
- $f(x)$ is large if the angle $\angle(w, x)$ is close to zero,
- it assigns large values to x with $\angle(w, x)$ close to zero,

Simplest neural network:

- $x = (x_0, x_1, \ldots, x_{d-1})^\top \in \mathbb{R}^d$ - input vector.
- $f(x)$ output of the only weight layer
- with weight vector $w = (w_0, w_1, \ldots, w_{d-1}, b) \in \mathbb{R}^d$
- $y = sign(z)$ - activation function on top of $f(x)$

Two types of classification problems:

Multi-class




exactly one of cat or mouse

Multi-label




none / cat / mouse / cat+mouse

Multi-class vs Multi-label classification

- A classification problem is multi-class, if for every sample exactly one ground truth class is present. The ground truth can be represented by a one-hot label.
- A classification problem is multi-label, if for every sample zero to C ground truth classes can be present. The ground truth can be represented by either the zero vector or a sum of one-hot labels.

- Most benchmark datasets are multiclass
- Most real-life classification problems are multi-label[2]
- the difference affects how to compute probabilities for the output, how to measure error, how to define training loss

[2] if one just needs to account for absence of anything, one can add a background class in multi-class setups

- multiclass: i-th one-hot vector. exactly one entry is one, all others are zero.

$$e^{(i)} = (0, \ldots, 0, \underbrace{1}_{i}, 0, \ldots, 0)$$

$$(1,0,0,0,0),\ (0,1,0,0,0),\ (0,0,1,0,0),\ (0,0,0,1,0),\ (0,0,0,0,1)$$

- multilabel: valid multilabel ground truths: (zero or sum of one-hot)

$$(0,0,0,0,0)$$
$$(0,1,0,0,0)$$
$$(0,0,1,0,1)$$
$$(1,1,1,1,1)$$