Visual AI

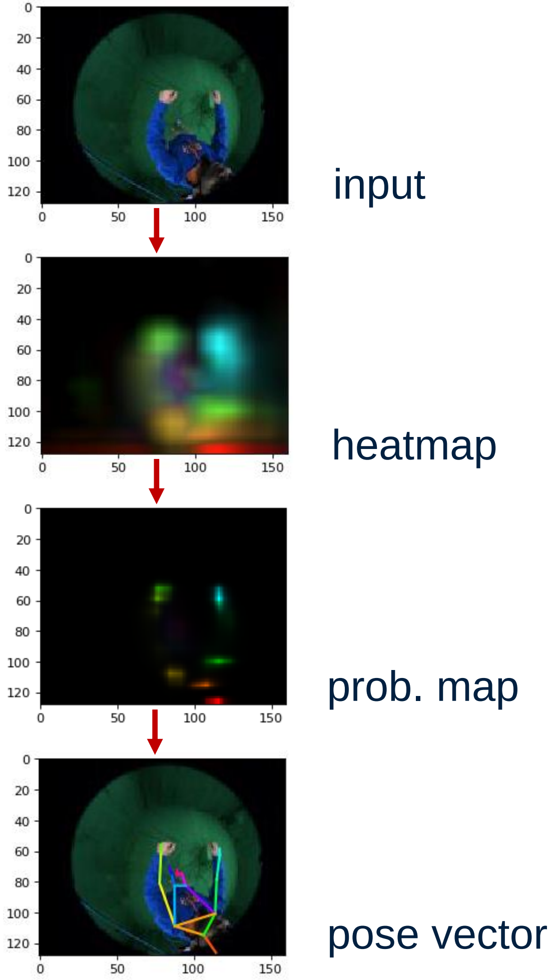
CPSC 533R

Lecture 4b. keypoints and probabilities

Helge Rhodin

Assignment II

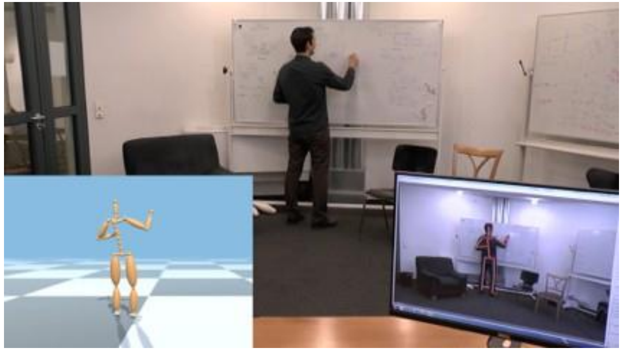
New teams formed today

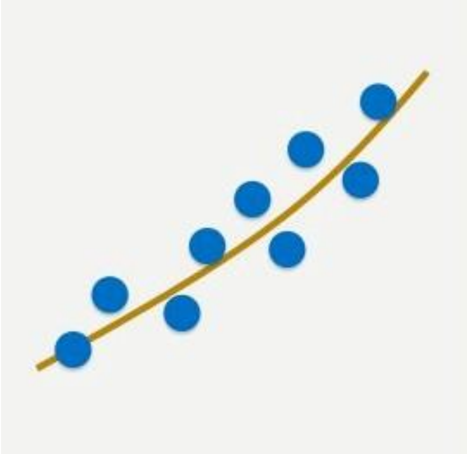


Classification vs. regression

Classification

Regression





Classification and regression

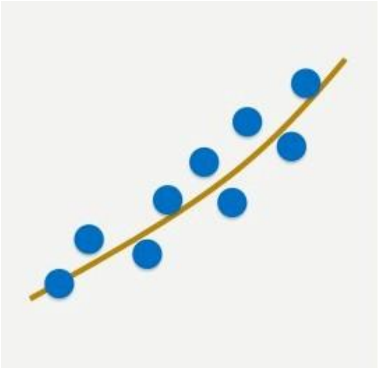
Regression

- for continuous values

$$\mathrm{nn}(\mathbf{x}) \rightarrow y \in \mathbb{R}$$

- squared loss is most common

$$l_2(y, l) = (y - l)^2$$



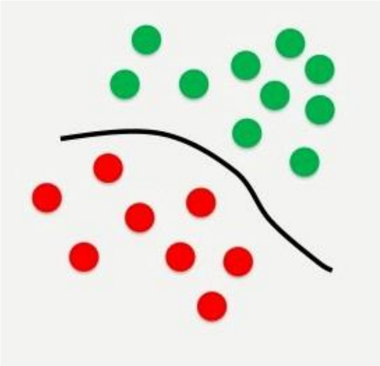
Classification

- discrete classes

$$\mathrm{nn}(\mathbf{x}) \rightarrow \mathbf{y} \in [0, 1]$$

- naïve least-squares loss

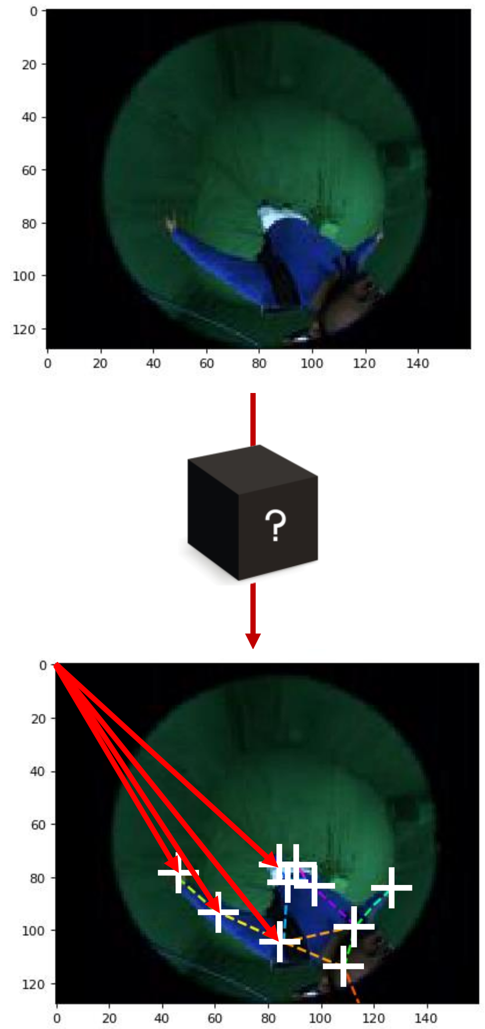
$$l_2(\mathbf{y}, \mathbf{l}) = \|\mathbf{y} - \mathbf{l}\|^2$$



Regression-based 2D pose estimation

A classical regression task

- Input:
  - grid of color values, an image (3 x W x H)
- Output:
  - pairs of continuous values, the position in the image
  - one pair for each of the K keypoints (2 x K)
- Neural network architecture:
  - Some convolutional layers to infer an internal representation of the human pose (C x W' x H')
  - One or more fully-connected layers to aggregate spatial information into the output values
    (C * W' * H') → (2 x K)

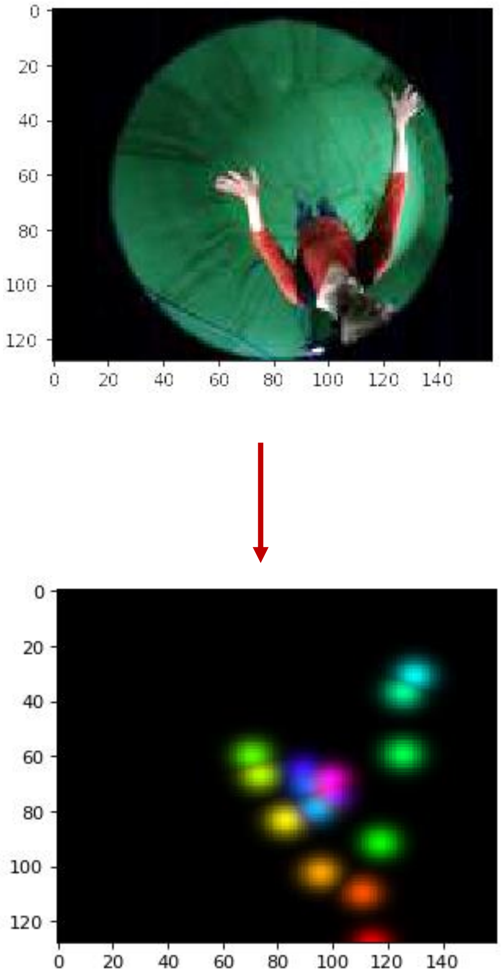


Binning

Heatmap-based 2D pose estimation

Phrase the regression task as classification

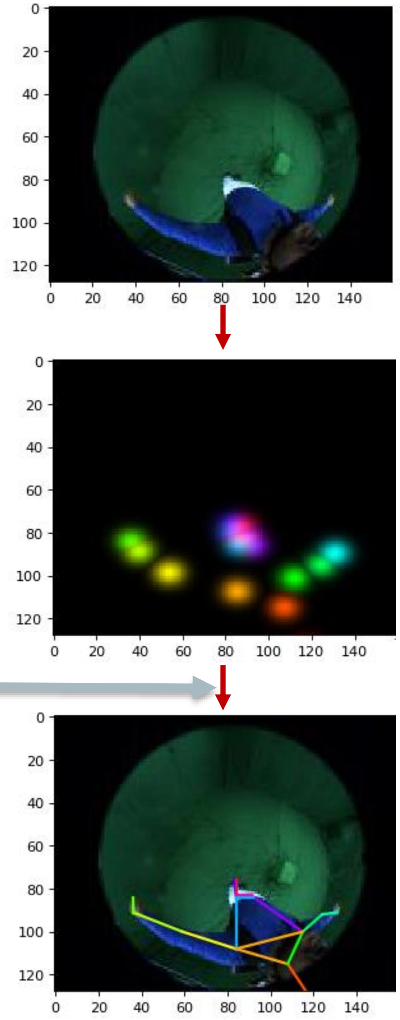
- separate heatmap H_j for each joint j
- Each pixel of H_j encodes the '*probability*' of containing joint j
  - not a true probability as pixels don't sum to one
- Advantages:
  - Inferred with fully convolutional networks
    - less parameters than fully connected ones (MLPs)
    - applies to arbitrary image resolution and aspect ratio (can be different from training)
    - translation invariance
    - locality
  - Generalizes to multiple and arbitrary number of persons



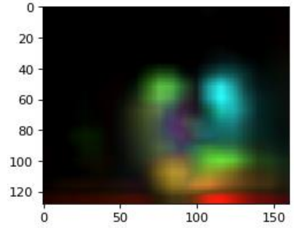
[Tompson et al., Efficient object localization using convolutional networks.]

Disadvantages of heatmaps

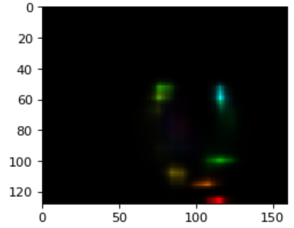
- Disadvantage:
  - Large image scale variations
    - Two-stage pipelines are alleviating this
      1. Detect person bounding box at coarse resolution
      2. Infer skeleton pose within box at high resolution
  - Not end-to-end differentiable (pose extraction requires arg-max function)
  - No sub-pixel accuracy
    - multi-scale approaches can overcome this at the cost of execution time (average over runs on re-scaled input)



Expectation of position

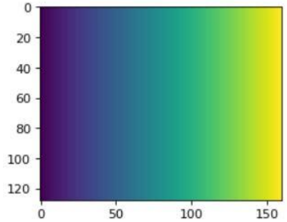


heatmap



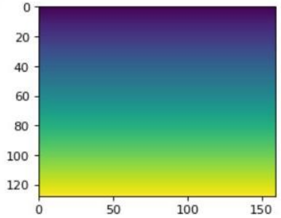
$p(x_i)$

probability map

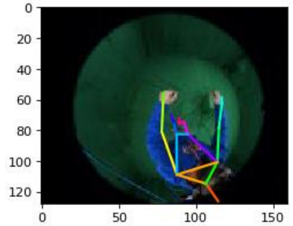


x_i

x-position



y-position



pose vector

Expectation (definition)

$$\mathrm{E}_{x \sim p}\, f(x) = \sum_{i=1}^{C} f(x_i) p(x_i)$$

Expectation of the x-position

$$\mathrm{E}_{x \sim p}\, x = \sum_{i=1}^{C} x_i p(x_i)$$

Integral Regression-based 2D pose estimation I

A combination of classification and regression

1. Detection network to produce heatmaps
   - same CNN as for heatmap prediction
2. Soft-max layer to turn heatmap H into probability map P
   - normalizing all pixels in each heatmap H

$$P[u,v] = \text{soft-max}(H,(u,v)) = \frac{e^{H[u,v]}}{\sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} e^{H[x,y]}}$$

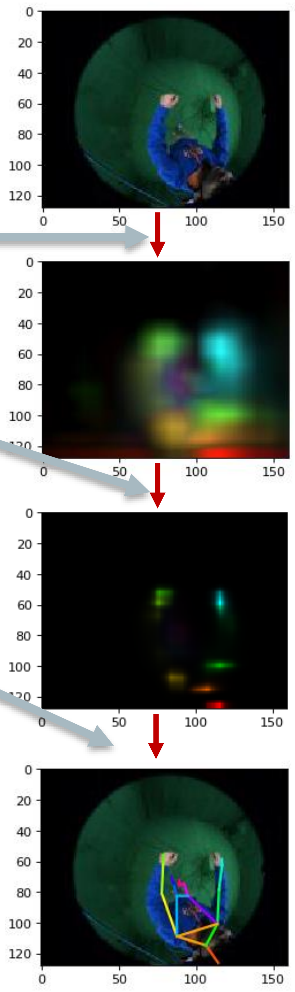
3. Integration layer to regress joint position (expected position)
   - can be interpreted as voting/weighted average

     each pixel votes for its own position, weighted by its probability

$$\text{pose}_x = \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} xP[x,y]$$

$$\text{pose}_y = \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} yP[x,y]$$

[Sun et al., Integral Human Pose Regression.]



input

heatmap

prob. map

pose vector

Integral Regression-based 2D pose estimation II

Advantages

1. Fully-convolutional CNN (as for heatmap classification)
2. Differentiable 2D pose regression
    - soft-max is differentiable, stable, and efficient to compute

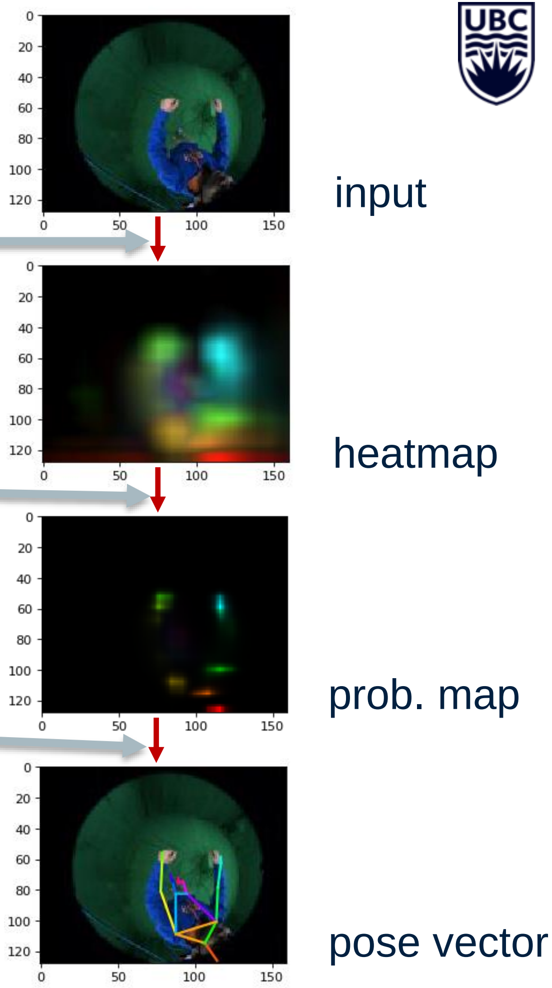
$$P[u,v] = \text{soft-max}(H,(u,v)) = \frac{e^{H[u,v]}}{\sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} e^{H[x,y]}}$$

    - sum over probability map is differentiable

$$\text{pose}_x = \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} x P[x,y]$$

$$\text{pose}_y = \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} y P[x,y]$$

3. End-to-end training
    - no difference between training and inference
    - sub-pixel accuracy possible through joint influence of pixels
        - low-resolution heatmaps possible



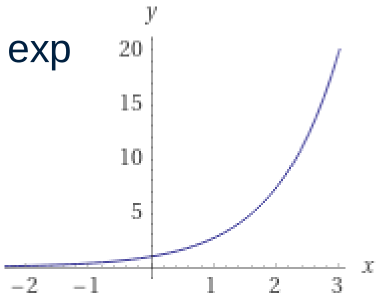
Attention: numerical stability

Exp normalize trick within cross-entropy

$$\text{soft-max}(z, i) = \frac{e^{z[i]} {\color{red}-\bar{z}} \; {\color{red}e^{\bar{z}}}}{\sum_{j=1}^{K} e^{z[j]} {\color{red}-\bar{z}} \; {\color{red}e^{\bar{z}}}}$$

$$= \frac{e^{z[i]-\bar{z}}}{\sum_{j=1}^{K} e^{z[j]-\bar{z}}}$$

exp



shift invariance is used to increase numerical stability!

The PyTorch implementation of cross-entropy includes this step

Issues?

Your laptop / desktop

- No GPU? -> google colab or university (see lecture 2)
- Note, parallel dataloaders might not work well on Windows:
  Error: “Can't pickle <function <lambda> …”
  - fix: disable threading by setting num_workers=0
- Other issues encountered?

Likelihood

The likelihood function measures the goodness of fit of a statistical model to a sample of data for given values of the unknown parameters

Sample data:

the N labels in the minibatch, input (images) x and labels c

Model:

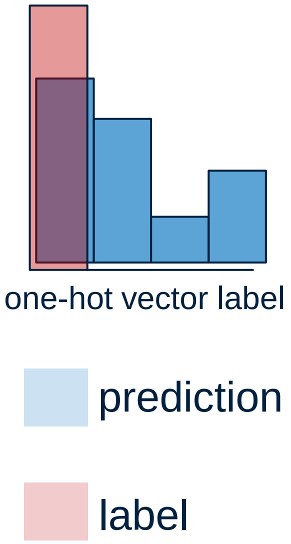
the neural network $f_\theta(\mathbf{x})$ defined by its architecture and parameters

Neural network output:

probabilities $f \in \mathbb{R}^N$ over the possible outcomes

Likelihood:

the probability $f_{[c]}(x)$ of the true class (the label)



Negative Log Likelihood (NLL) for a *one-hot vector*

$$l_{\text{NLL}}(x, c) = -\log(f_{[c]}(x))$$

subscript [c] denotes the c'th value of output vector

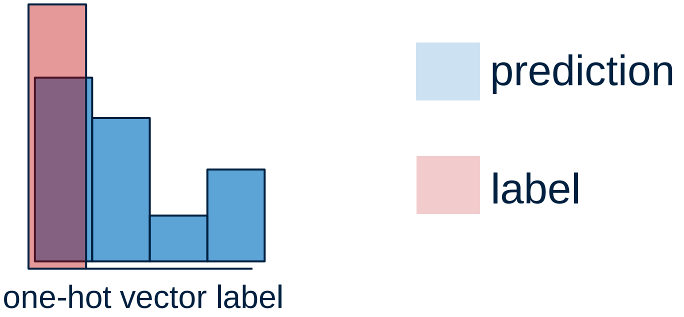
The function f must output a distribution/PMF (sum to 1)!

Cross-entropy loss / Cross-entropy criterion

Negative Log Likelihood (NLL) for a *one-hot vector* $f \in \mathbb{R}^N$,
with c the ground truth target class

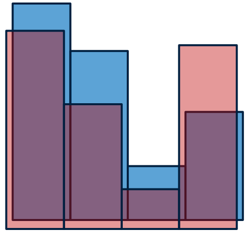
$$l_{\text{NLL}}(x, c) = -\log(f_{[c]}(x))$$

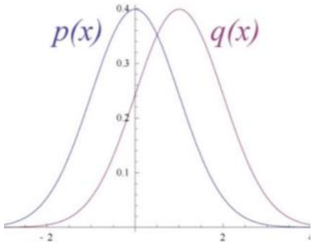
subscript [c] denotes the c'th value of output vector



Cross entropy definition:

$$H(p, q) = -\mathrm{E}_p[\log q] = -\sum_{c=1}^{K} p(c) \log q(c)$$



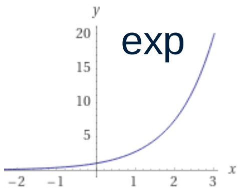


Cross-entropy loss for label y

$$l_{\text{cross entropy}}(x, y) = -\sum_{j=1}^{K} y_{[j]} \log(f_{[j]}(x))$$

The function f must output a distribution (sum to 1)!

Cross correlation with a soft-max layer



Negative log likelihood with preceding soft-max (log-soft-max)

$$l_{\text{log-likelihood}}(x, y) = -\log(\text{soft-max}(f(x), y))$$

$$= -f_{[y]}(x) + \log\left(\sum_{j=1}^{K} e^{f_{[j]}(x)}\right)$$

Soft-max

$$\text{soft-max}(z, i) = \frac{e^{z[i]}}{\sum_{j=1}^{K} e^{z[j]}}$$

log-sum-exp trick (for log-soft-max)

$$\text{log-sum-exp}(z) = \log\left(\sum_{j=1}^{K} e^{z}\right)$$

$$= \bar{z} + \log\left(\sum_{j=1}^{K} e^{z-\bar{z}}\right)$$

$$\text{with } \bar{z} = \max(z)$$

exp-normalize trick (for soft max)

$$\text{soft-max}(z, i) = \frac{e^{z[i]-\bar{z}} \; e^{\bar{z}}}{\sum_{j=1}^{K} e^{z[j]-\bar{z}} \; e^{\bar{z}}}$$

$$= \frac{e^{z[i]-\bar{z}}}{\sum_{j=1}^{K} e^{z[j]-\bar{z}}}$$

shift invariance is used to increase numerical stability!

Cross-entropy loss in PyTorch

```
def def cross_entropy(input, …
        return nll_loss(log_softmax(input …
```

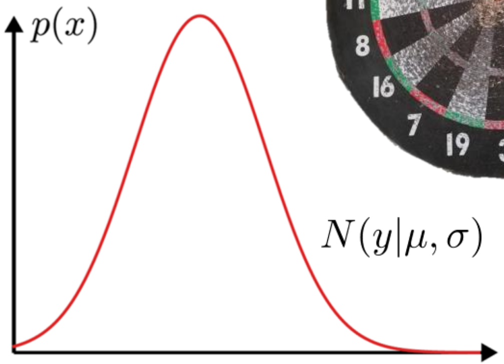
- Includes the normalization by log-soft-max
  - numerically stable
  - fast
  - don’t normalize twice with your own soft-max layer followed by cross_entropy!

Probabilistic interpretation of least squares regression

Regression: minimize the negative log-likelihood

Likelihood:

$$N(y|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

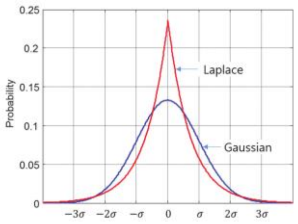


Negative log likelihood:

(simplified)

$$E = \frac{1}{2\sigma^2}(y - f_\theta(x))^2$$

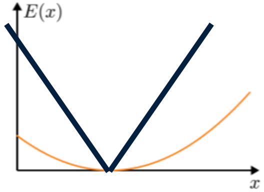
Likelihood



Gaussian distribution $\exp(-x^2)$

Laplace distribution $\exp(-|x|)$

Error functions (negative log likelihood)



Mean squared error (MSE) x^2

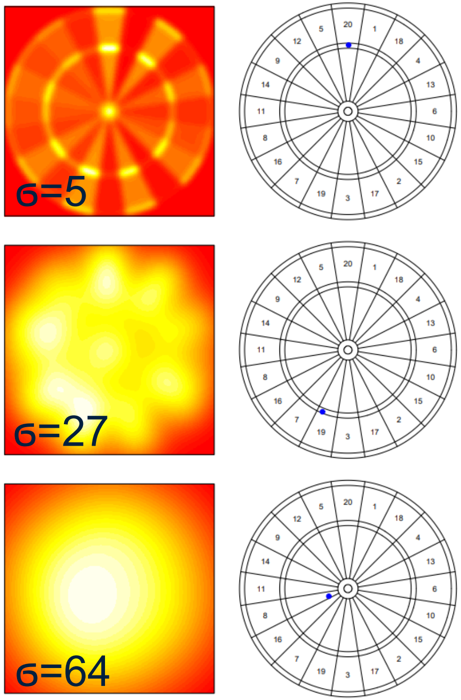
Mean absolute error (MAE) $|x|$

Darts

“… Assuming standard scoring, **the optimal area to aim for on the dartboard to maximize the player's score varies significantly based on the player's skill**. The skilled player should aim for the centre of the T20, and as the player's skill decreases, their aim moves slightly up and to the left of the T20. At σ = 16.4 mm the best place to aim jumps to the T19. As the player's skill decreases further, the best place to aim curls into the centre of the board, stopping a bit lower than and to the left of the bullseye at σ = 100.[28]
Where σ may refer to the standard deviation for a specific population.”

https://en.wikipedia.org/wiki/Darts

[28] Ryan J. Tibshirani, Andrew Price, and Jonathan Taylor (January 2011) ["A statistician plays darts"]() [Archived]() 2011-07-20 at the [Wayback Machine](), *Journal of the Royal Statistical Society*, series A, vol. 174, no. 1, pages 213–226



Left: Expected score when aiming at a certain location.
Right: its maximum

Probabilistic interpretation of least squares regression

Regression: minimize the negative log-likelihood (here equal to the MSE)

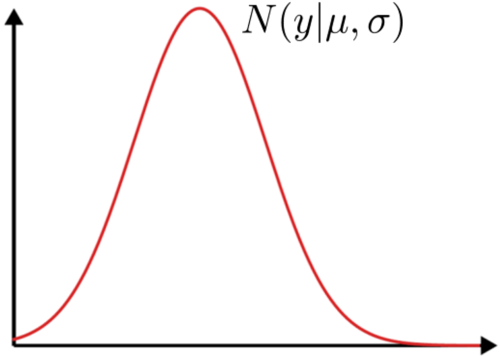
$$E = \frac{1}{2\sigma^2}(y - f_\theta(x))^2$$

Density Networks: Predict the mean μ **and standard deviation σ**

$$\sigma, \mu = f_\theta(x) \qquad \text{of the likelihood} \qquad N(y|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

Minimize the negative log-likelihood (now treating the std. dev. as a parameter)

$$\begin{aligned} -\log(L) &= -\log(N(y|\mu,\sigma)) \\ &= -\log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \frac{(y-\mu)^2}{2\sigma^2} \\ &= -\log\left(\frac{1}{\sigma_\theta(x)\sqrt{2\pi}}\right) + \frac{(y-\mu_\theta(x))^2}{2\sigma_\theta(x)^2} \end{aligned}$$



Self-study: Probability theory

Cheat sheet: computing expectations

For random events/variables we can only reason about the expected outcome and its estimates over a finite set of samples

Discrete set of C classes:

Definition

$$\mathrm{E}_{x\sim p}\, f(x) = \sum_{i=1}^{C} f(x_i)p(x_i)$$

Continuous distribution:

Definition

$$\mathrm{E}_{x\sim p}\, f(x) = \int_{\Omega} f(x)p(x)\, dx$$

Estimators for discrete classes

Empirical estimate

$$\mathrm{E}_{x\sim p}\, f(x) \approx \frac{C}{N}\sum_{i=1}^{N} f(x_i) \text{ with } x_i \sim p$$

Uniform Monte Carlo sampling

$$\mathrm{E}_{x\sim p}\, f(x) \approx \frac{C}{N}\sum_{i=1}^{N} f(x_i)p(x_i)$$

with N samples x_i drawn uniformly at random

Importance sampling

$$\mathrm{E}_{x\sim p}\, f(x) \approx \frac{C}{N}\sum_{i=1}^{N} \frac{p(x_i)}{q(x_i)} f(x_i) \text{ with } x_i \sim q$$

Basic definitions: Discrete random variables

The Probability Mass Function (PMF) gives the probability that a discrete random variable X takes on the value x.

$$p_X(x) = P(X = x)$$

The PMF satisfies

$$p_X(x) \geq 0 \text{ and } \sum_x p_X(x) = 1$$

Expected Value (a.k.a. mean, expectation, or average) is a weighted average of the possible outcomes of our random variable.

$$E(X) = \sum_i x_i P(X = x_i)$$

Random variable? A variable whose values depend on outcomes of a random phenomenon. E.g. measurement noise or uncertainty when predicting the future.

Basic definitions: Continuous random variables

The probability density function (**PDF**), short: **density** $f_X(x)$ of a continuous random variable *X*, is a function whose value at any given sample point provides a relative likelihood that the value of the random variable would equal that sample. It holds

$$\Pr[a \leq X \leq b] = \int_a^b f_X(x)\, dx.$$

Note, $P[X = a] = 0$ but $f_X(x) \neq 0$
(infinitely many possible outcomes, each individual has mass 0)

How do I find the expected value for continuous events? Analogous to the discrete case, where you sum x times the PMF, now you integrate

$$E(X) = \int_{-\infty}^{\infty} x f(x) dx$$

Likelihood function: The likelihood is simply the PDF regarded as a function of the parameter rather than of the data. It is no longer a pdf with respect to the parameters theta