Machine Learning

8. Information theory

Yannick Le Cacheux

CentraleSupélec - Université Paris Saclay

September 2024

Outline

1 10. Information theory

2 Curse of dimensionality

3 t-SNE

Can we quantify information?

- Suppose we observe a specific value of a random variable x sampled from a discrete distribution p
- "How much" do we learn? How much information is received?
- Intuitively, we expect to be more surprised, *i.e.* to receive more information, if we observe a very unlikely event than if we observe an expected event

Defining the amount of information

- We will write $h(x)$ the quantity we are looking for, *i.e.* the amount of information received when observing x
- We are looking for a quantity $h(x)$ decreasing when $p(x)$ increases
- For events x that were guaranteed to happen, *i.e.* such that $p(x) = 1$, we don't learn anything so $h(x) = 0$

Defining the amount of information

- If we observe two unrelated events x and y, we learn the sum of the corresponding information, so that

$$h(x, y) = h(x) + h(y)$$

- For independent events x and y, we have

$$p(x, y) = p(x)p(y)$$

- So we can actually define

$$h(x) = -\log p(x)$$

Amount of information

- We call $h(x)$ the *amount of information*:

$$h(x) = -\log p(x)$$

- We have $h(x) \geq 0$
- By convention, we will often express $h(x)$ with a $\log_2$ so that the result is the amount of information in *bits*
- But we could also use $\ln$, in which case the result is the amount of information in *nats*

Entropy

- Given a probability distribution p with outcomes in $\mathcal{X}$, the expected information h is

$$\begin{aligned}\mathbb{E}[h] &= \sum_{x \in \mathcal{X}} p(x) h(x) \\ &= -\sum_{x \in \mathcal{X}} p(x) \log p(x) \\ &= H[x] \quad \text{(entropy of random variable } x\text{)} \\ &= H[p] \quad \text{(entropy of distribution } p\text{)}\end{aligned}$$

[0]Note: here, since p is a discrete probability distribution, we would usually rather write P to denote the corresponding probabilities, as we would rather use p to denote a continuous *probability density function*. However, to keep notations consistent with the illustrations I am using, p can be either discrete probabilities or a continuous probability density function here.

Entropy of a Bernoulli distribution

- Example: a Bernouilli distribution $\mathcal{B}_\rho$ with parameter ρ is defined such that

$$P(x=1) = \rho \qquad P(x=0) = 1-\rho$$

$$\text{so} \quad H[\mathcal{B}_\rho] = -\sum_{x \in \{0,1\}} P(x) \log P(x) = \rho \log \rho + (1-\rho)\log(1-\rho)$$

Thus,

$$H[\mathcal{B}_{0.9}]$$
$$= -(0.1 \log_2(0.1) + 0.9 \log_2(0.9))$$
$$\approx 0.47$$

$$H[\mathcal{B}_{0.5}] = 1.00$$
$$H[\mathcal{B}_0] = 0.00$$

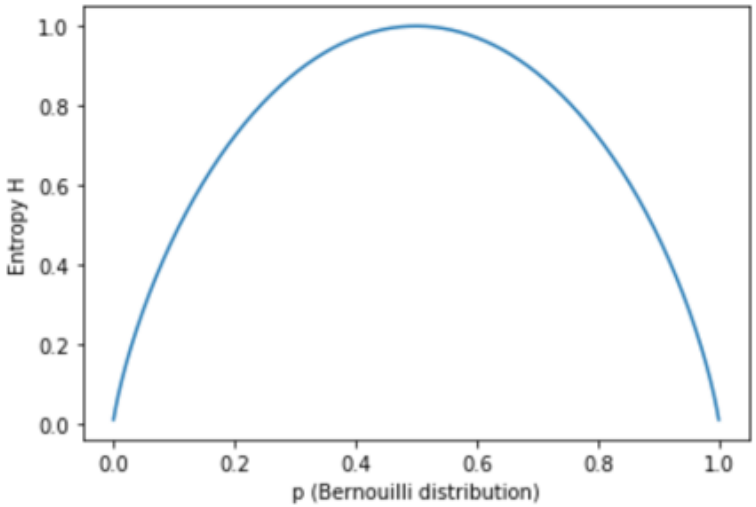


Figure: Entropy for Bernoulli distribution $\mathcal{B}_\rho$ for different ρ

Shannon's theorem

- Entropy can be defined as the "average amount of information needed to specify the state of a random variable" [1]

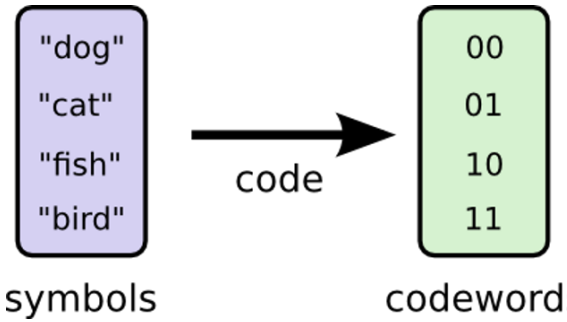
Shannon's source coding theorem

The average number of bits needed to transmit the state of a random variable is at least the entropy of the corresponding probability distribution

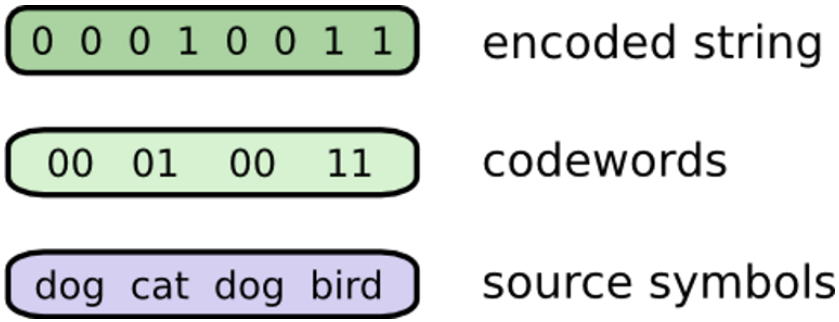
- Entropy can be viewed as the (minimum) number of bits necessary to transmit information regarding values sampled in a given distribution
- It is also a measure of disorder, thus being related to the entropy used in physics

[1] C. Bishop, *Pattern recognition and machine learning*

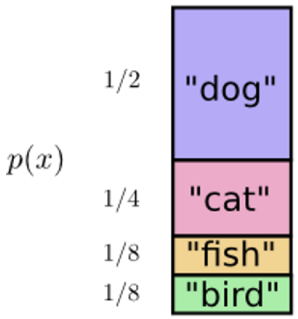
Entropy in communication



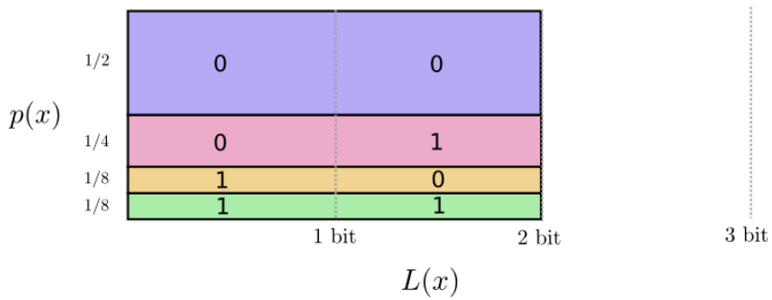
(a) Symbols to codewords



(b) Example encoded message



(a) Dog lover's word frequency[2]

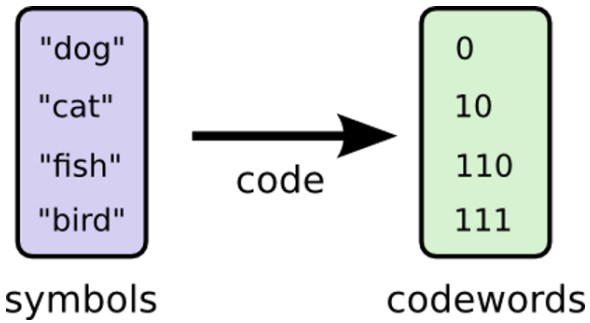


(b) Entropy of codewords: 2 bits

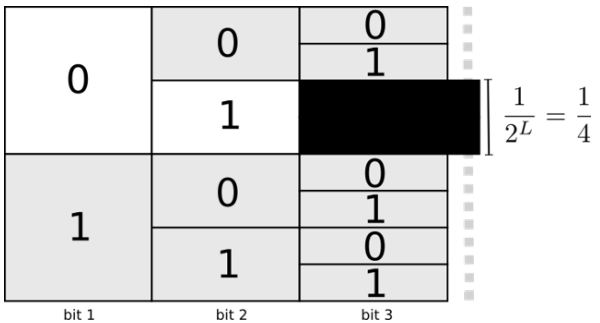
- We can communicate symbols using codewords
- Is this scheme optimal?

[2] Illustrations from colah.github.io, as are all similar figures in this section.

Entropy in communication



(a) Better codewords



(b) Codewords space cost

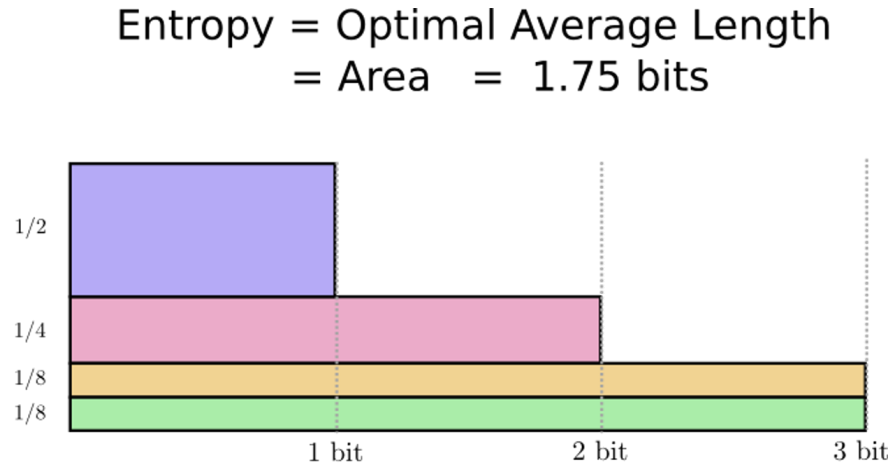


Figure: Resulting entropy

$$
\begin{aligned}
H &= \mathbb{E}[\text{Code}_{\text{length}}] = \sum_x p(x) \log_2 \left(\frac{1}{p(x)} \right) \\
&= -\Big(\frac{1}{2} \log_2(\frac{1}{2}) + \frac{1}{4} \log_2(\frac{1}{4}) + \\
&\frac{1}{8} \log_2(\frac{1}{8}) + \frac{1}{8} \log_2(\frac{1}{8}) \Big) \\
&= 1.75
\end{aligned}
$$

Joint entropy

- Let us consider a joint probability distribution $p(x, y)$
- The corresponding joint entropy is

$$H[x, y] = -\sum_x \sum_y p(x, y) \log p(x, y)$$

- We sample x and y from this distribution
- If we know x, the average amount of information needed to specify y is

$$-\log p(y|x)$$

- So the expected additional information needed is

$$H[y|x] = -\sum_x \sum_y p(x, y) \log p(y|x)$$

Conditional entropy

- $H[y|x]$ is the conditional entropy of y given x

$$H[y|x] = -\sum_x \sum_y p(x,y) \log p(y|x)$$

- If we write $H[x,y]$ the entropy of the joint distribution, we have

$$H[x,y] = H[y|x] + H[x]$$

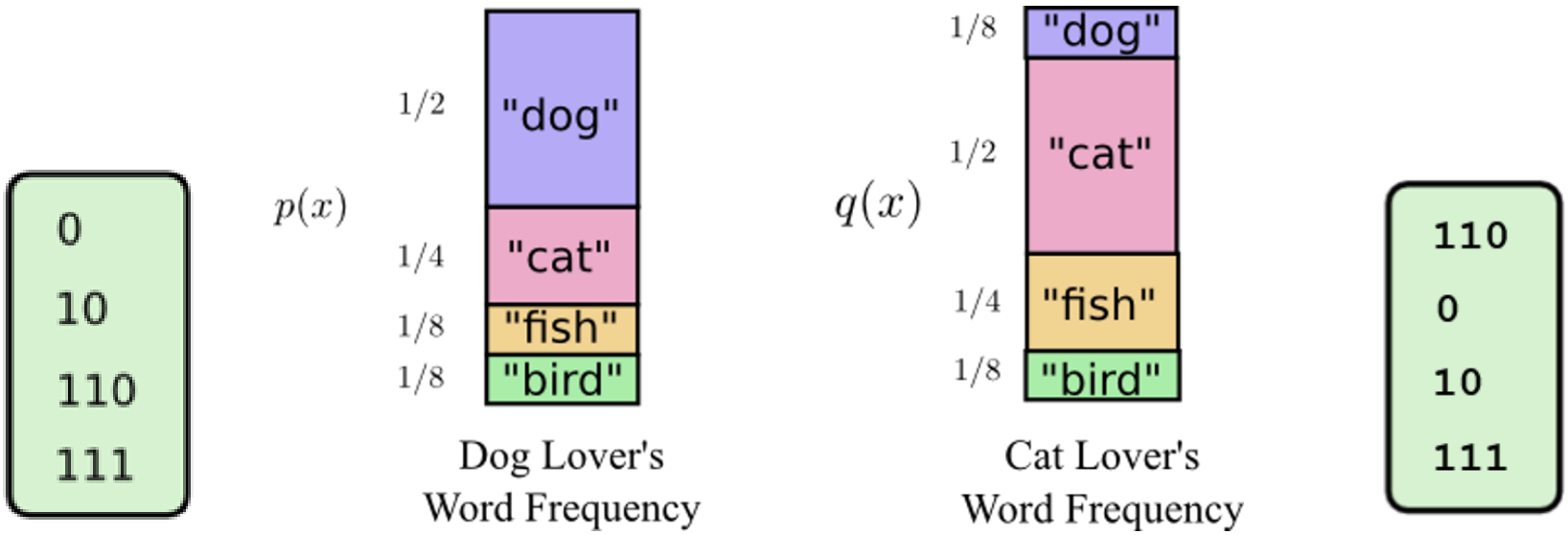
  - If we know x, we need $H[y|x]$ additional bits to express y

- In general, $H[x,y] \leq H[x] + H[y]$, with equality if and only if x and y are independent[3]

[3] For a proof, *cf.* homework assignment 2

Cross entropy

- Now let us consider 2 distinct distributions p and q, with corresponding codewords



- It is not optimal to use the codewords from q to encode words from p

Cross entropy

- The cross-entropy $H_q[p]$ of p w.r.t. q can be intuitively defined as the average length of a message from distribution p with the optimal "code" from q

$$\begin{aligned} H_q[p] &= \mathbb{E}_p[-\log q] \\ &= -\sum_{x \in \mathcal{X}} p(x) \log q(x) \end{aligned}$$

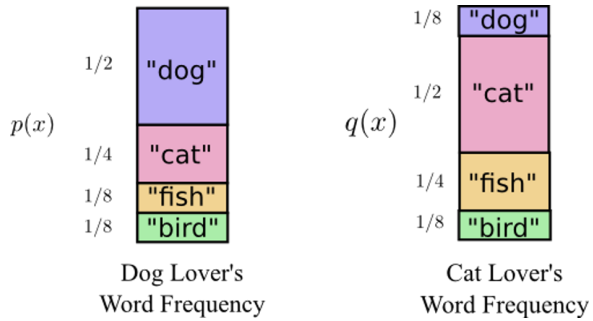
(sometimes also written $H[p, q]$, e.g. on Wikipedia)

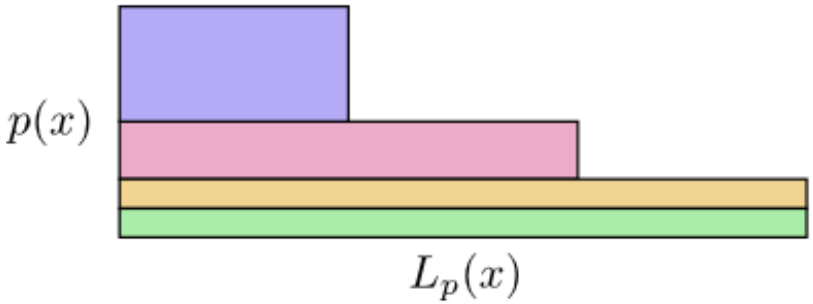
- This is the same thing as the cross entropy loss $\mathcal{L}_{\text{ce}}$ we use in deep learning, or the logistic loss: for K classes,

$$\mathcal{L}_{\text{ce}}(\mathbf{y}, \hat{\mathbf{y}}) = -\underbrace{\frac{1}{N} \sum_{n=1}^{N}}_{\mathbb{E}} \underbrace{\sum_{k=1}^{K}}_{x \in \mathcal{X}} \underbrace{\mathbb{1}[y_n = k]}_{\text{approximation of } p(x)} \log\Big(\underbrace{\hat{y}_k}_{\text{estimated } q(x)} \Big)$$

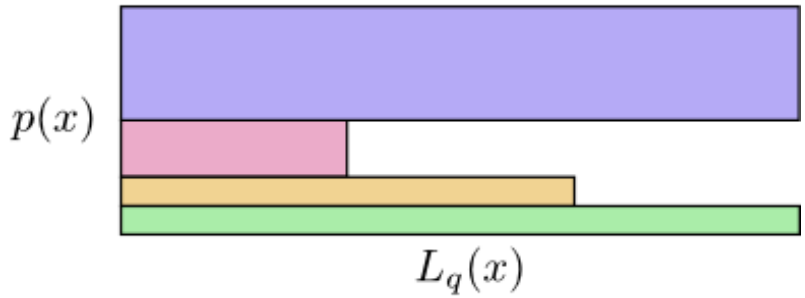
Cross entropy

In general, $H_q[p] \neq H_p[q]$

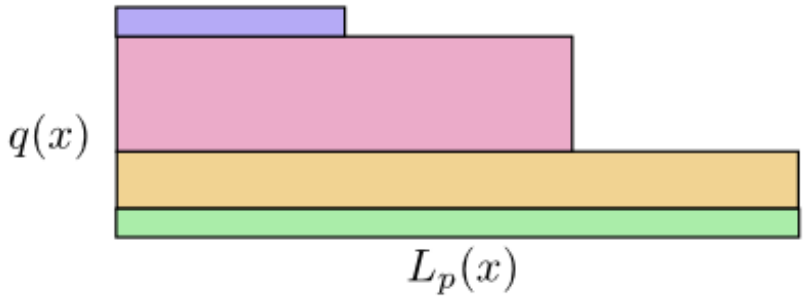




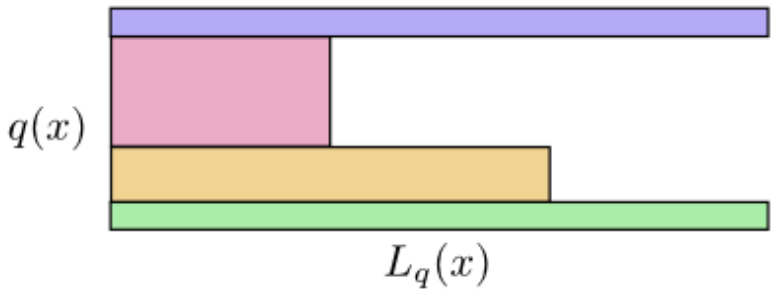
(a) $H[p] = H_p[p] = 1.75$bits



(b) $H_q[p] = 2.375$bits



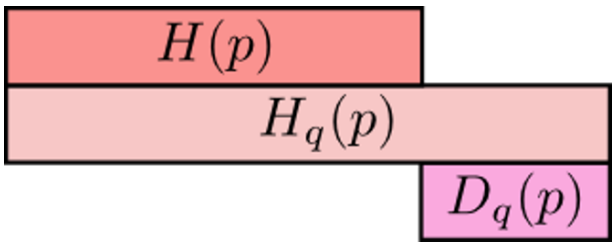
(c) $H_p[q] = 2.25$bits



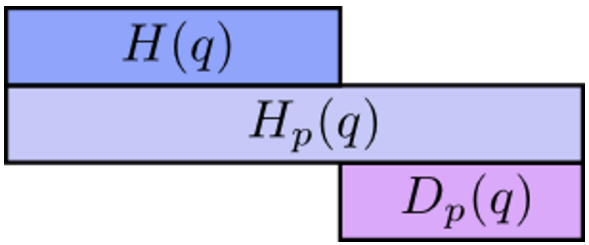
(d) $H[q] = H_q[q] = 1.75$bits

Cross entropy as a measure of similarity

- Cross-entropy gives us an idea of how close distribution q is to distribution p
- If q is close to p, the increase in entropy from encoding p with codewords for q will not be too large



(a) Additional entropy $D_q[p]$ needed when using optimal codewords for q to encode messages from p



(b) Additional entropy $D_p[q]$ needed when using optimal codewords for p to encode messages from q

- In logistic regression / with a cross entropy loss, we want the estimated distribution to be close to the true distribution of the data

Kullback-Leibler divergence

- The difference $D_q[p]$ between $H_q[p]$ and $H[p]$ is called the Kullback-Leibler divergence $D_{\mathsf{KL}}[p\|q]$ of p w.r.t. q

$$
\begin{aligned}
D_{\mathsf{KL}}[p\|q] = D_q[p] &= H_q[p] - H[p] \\
&= -\sum_x p(x) \log q(x) - \left(-\sum_x p(x) \log p(x) \right) \\
&= -\sum_x p(x) \log \frac{q(x)}{p(x)} \qquad \left(= \sum_x p(x) \log \frac{p(x)}{q(x)} \right)
\end{aligned}
$$

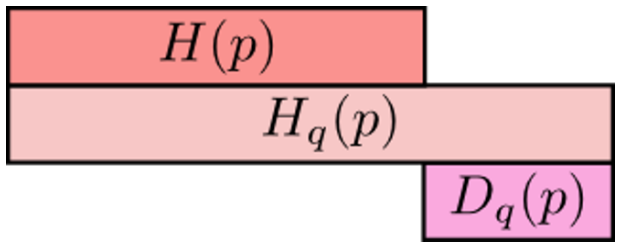
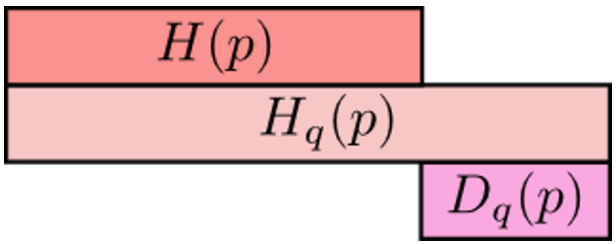


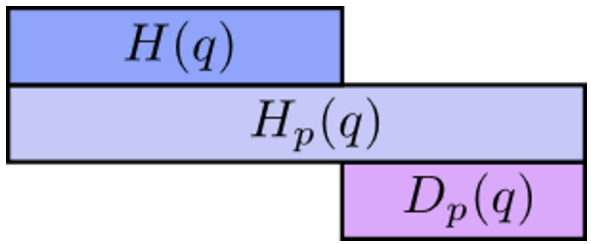
Figure: Difference $D_q[p]$ between $H_q[p]$ and $H[p]$

Kullback-Leibler divergence in communication

- $D_{\mathrm{KL}}[p\|q]$ is the additional quantity of information required to specify the state of x if we use $q(x)$ as a "coding scheme" as opposed to using the optimal scheme $p(x)$



(a) Additional entropy $D_q[p]$ needed when using optimal codewords for q to encode messages from p



(b) Additional entropy $D_p[q]$ needed when using optimal codewords for p to encode messages from q

Kullback-Leibler divergence

- Is $D_{\mathsf{KL}}[p][p\|q]$ a distance between p and q?
  - We have $D_{\mathsf{KL}}[p\|q] \geq 0$
  - $D_{\mathsf{KL}}[p\|q] = 0 \iff p = q$
  - But in general $D_{\mathsf{KL}}[p\|q] \neq D_{\mathsf{KL}}[q\|p]$

Mutual information

- Let us consider a pair of random variables x and y with respective distributions p_x and p_y, and joint distribution $p_{x,y}$
- The *mutual information* I between x and y is

$$I(x, y) = D_{\mathsf{KL}}(p_{x,y} \| p_x p_y)$$

- It quantifies the amount of information obtained about one random variable by observing the other random variable.

$$I(x, y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{x,y}(x, y) \log \frac{p_x(x) p_y(y)}{p_{x,y}(x, y)}$$

Differential entropy

- We initially defined entropy H as the expected amount of information h from distribution p:

$$\begin{aligned} H[p] &= \mathbb{E}_p[h] \\ &= \sum_{x \in \mathcal{X}} p(x) h(x) \quad \text{for a discrete distribution } p \\ &= -\sum_x p(x) \log p(x) \end{aligned}$$

- Using this definition, entropy can be generalized to multi-variable (joint) probability distributions $p(\mathbf{x})$, and to continuous probability density functions:

$$H[p] = -\int_{\mathbf{x} \in \mathbb{R}^D} p(\mathbf{x}) \log p(\mathbf{x}) \mathrm{d}\mathbf{x}$$

- The continuous version is called the *differential entropy*

Kullback-Leibler in general

- Similarly to entropy, the Kullback-Leibler divergence can be expressed as an expectation:

$$KL[p\|q] = \mathbb{E}_p \left[-\log \left(\frac{q}{p} \right) \right]$$

- Which yields the Kullback-Leibler divergence for continuous probability density functions:

$$KL[p\|q] = -\int p(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} \mathrm{d}\mathbf{x}$$

- Same idea with cross-entropy, joint entropy etc

Differential entropy

- There are a few notable differences between entropy and differential entropy: for instance, differential entropy can be negative
- Noteworthy result: on a finite-size interval, the function p which maximizes the differential entropy[4] is uniform
  - Intuitively, this makes sense: earlier, if we did not know the parameter ρ of a Bernouilli distribution $\mathcal{B}_\rho$, we may assume that $\rho = \frac{1}{2}$: this is the parameter maximizing the entropy
  - Similarly, if we do not know anything about a p.d.f. p defined on an interval, we may as well assume that it is constant
- Differential entropy can also be measured on infinite-size intervals
  - Related question: why do we often use the normal distribution "by default" in machine learning and other fields?

[4] We still need to have $\int_{[a,b]} p \, dx = 1$ etc so that p corresponds to a probability density function

Why use the normal distribution?

One possible answer:

Central limit theorem

The sum of many independent random variables tends toward a normal distribution

- In ML, there are typically many unobserved factors affecting our target variable
  - Features we did not collect
  - Hidden variables we cannot measure
  - Random noise from various sources (measurement error etc)
- The combined effect of all these unknown factors tends to follow a normal distribution

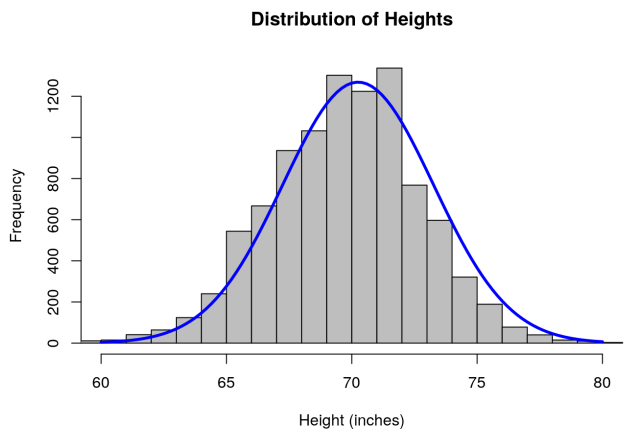


Figure: The height of adults depends on many independent genes

Differential entropy

Another possible answer:

- If we add the constraints

$$\int_{-\infty}^{\infty} p(x)dx = 1$$

$$\int_{-\infty}^{\infty} xp(x)dx = \mu$$

$$\int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx = \sigma^2$$

- We find[5] that the distribution p maximizing $H[p]$ is

$$p(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

- This is the normal distribution!

[5] Using variational calculus and Lagrange multipliers

Applications of information theory in machine learning

- We have already met applications in
  - Decision tree: entropy as a splitting criterion
  - Logistic regression: logistic loss is a special case of cross-entropy when there are 2 possible outputs
- Quantifying the similarity between probability distributions with Kullback-Leibler has many applications:
  - t-SNE: minimize KL between a high-dim distribution and a low-dim distribution
  - Variational inference: approximate an unknown distribution by a known distribution
- Mutual information can be used to select relevant features
- Perplexity (based on entropy) is used to evaluate language models
- ...

Outline

1 10. Information theory

2 Curse of dimensionality

3 t-SNE

Hyperspheres and hypercubes

- Let's consider a sphere of radius $R = 1$ inscribed in a cube of side $2R$.

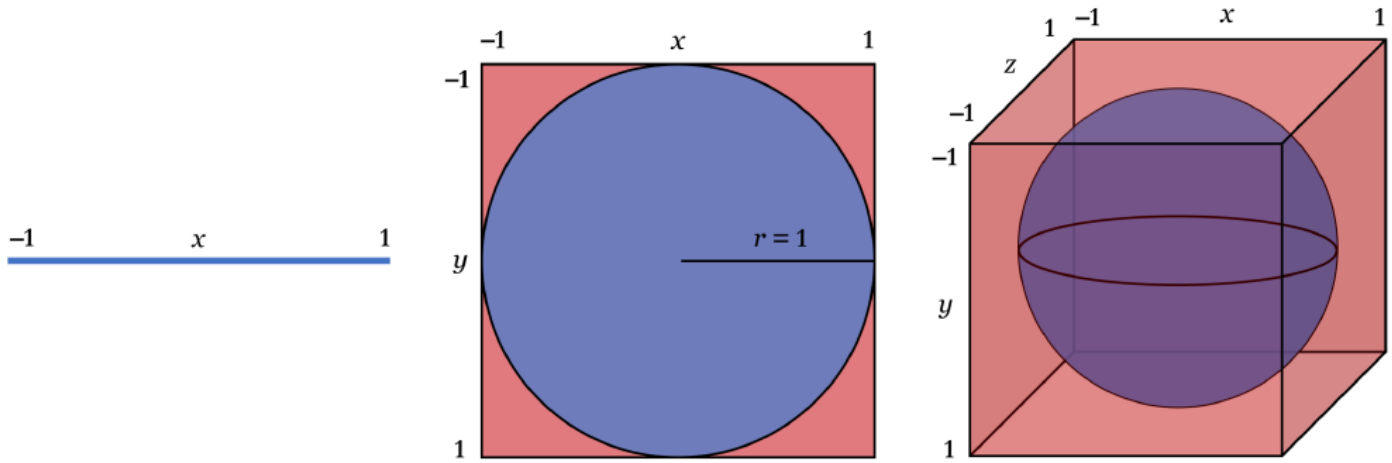


Figure: Inscribed sphere[6] in 1, 2 and $3D$

Ratio $V_{s/c}$ of the (hyper)sphere volume to the (hyper)cube:

- In $1D$: $V_{s/c} = 1.0$
- In $2D$: $V_{s/c} = 0.785$
- In $3D$: $V_{s/c} = 0.524$
- ...
- In $10D$: $V_{s/c} = 0.00249$
- In $100D$: $V_{s/c} = 2.38 \cdot 10^{-40}$

[6] Image from T. Rainford, *Lecture 5: Advanced Bayesian Inference Methods, Advanced Topics in Machine Learning, Oxford*.

Hyperspheres and hypercubes

$$V_{\text{sphere}}(D) = \frac{\pi^{\frac{R}{2}}}{\Gamma(\frac{R}{2}+1)} \quad V_{\text{cube}}(D) = 2^D$$

$$\text{so } V_{s/c} = \frac{V_{\text{sphere}}}{V_{\text{cube}}} \underset{D\to\infty}{\to} 0$$

- Assuming points are uniformly sampled in the hypercube: as D grows, almost no point is in the center.

Hyperspheres and hypercubes

$$V_{\text{sphere}}(D) = \frac{\pi^{\frac{R}{2}}}{\Gamma(\frac{R}{2}+1)} \quad V_{\text{cube}}(D) = 2^D$$

$$\text{so } V_{s/c} = \frac{V_{\text{sphere}}}{V_{\text{cube}}} \underset{D\to\infty}{\to} 0$$

- Assuming points are uniformly sampled in the hypercube: as D grows, almost no point is in the center.
- But almost no point is in a corner either.

- What the heck is going on?

Distances in large D

Distance d between points $\mathbf{a} = (a_1, \ldots, a_D)^T$ and $\mathbf{b} = (b_1, \ldots, b_D)^T \in \mathbb{R}^D$

- In $1D$:

$$d(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_2^2 = \sqrt{(a_1 - b_1)^2} = |a_1 - b_1|$$

- In $2D$:

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

- In D dimensions:

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_1 - b_1)^2 + \cdots + (a_D - b_D)^2}$$

Central Limit Theorem

The (normalized) sum of i.i.d. random variables with finite mean and variance converges to a normal distribution as the number of variables increases.

Assuming coordinates are i.i.d., distances thus tend to be normally distributed in high dimension.

Distances in large D

Central limit theorem

Distances tend to be normally distributed in high D.

- For the squared distance[7] to the origin d^2, assuming points are uniformly sampled in centered hypercube in D dimensions:

$$\mathbb{E}[d^2] = \frac{D}{3} = \mathcal{O}(D) \qquad \sqrt{\text{var}}[d^2] = \frac{2\sqrt{D}}{45} = \mathcal{O}(\sqrt{D})$$

- The standard deviation is growing slower than the mean. Thus, for large D, the (squared) distance d^2 normalized by D is close to a normal distribution with parameters:

$$\frac{d^2}{D} \sim \mathcal{N}(\frac{1}{3}, \frac{2}{\sqrt{45D}})$$

- Points are concentrated on the edge of a $\sqrt{d/3}$-radius hypersphere

[7] Results for d (non-squared) could also be derived with much more effort

Distances in large D

- In large D, points tend be concentrated on the edge of a hypersphere with radius $\sqrt{d/3}$
- Worse: for N points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and a reference point $\mathbf{x}_0$, define

$$d_{\min} = \min_n d(\mathbf{x}_0, \mathbf{x}_n) \qquad d_{\max} = \max_n d(\mathbf{x}_0, \mathbf{x}_n)$$

Then

$$\lim_{D \to \infty} \mathbb{E}\left[\frac{d_{\max} - d_{\min}}{d_{\max}}\right] = 0$$

- Distances tend to become meaningless in high dimension, converging all to a similar value[8]

[8] Again, this is assuming coordinates are uniformly sampled

Curse of dimensionality

Curse of dimensionality

Some words

- This has implications on many machine learning algorithms
  - For instance, K-Nearest Neighbors directly relies on distance, and may not be relevant in high D
- In practice, coordinates are not sampled uniformly and independently
  - But irrelevant features may still introduce significant noise that may drown out the actual signal

An additional word of caution

Our intuition on what happens in high dimensions tends to rely on parallels with low-dim phenomena, and is typically pretty bad

Outline

1 10. Information theory

2 Curse of dimensionality

3 t-SNE

Multidimensional scaling (MDS)

- Given: N points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$ in high-dim space
- (Dis)similarity functions $d : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^+$ (in high-dim space) and $d' : \mathbb{R}^H \times \mathbb{R}^H \to \mathbb{R}^+$ (in low-dim space)
- MDS attempts to find low-dim representations $\mathbf{z}_1, \ldots, \mathbf{z}_N \in \mathbb{R}^H$, $H < D$, that minimize *stress*, *i.e.* keep distances in high and low-dim spaces as close as possible:

$$\underset{\mathbf{z}_1,\ldots,\mathbf{z}_N}{\text{minimize}} \sqrt{\sum_{\substack{i,j \\ i \neq j}} \left(d(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{x}_i, \mathbf{x}_j)\right)^2}$$

- In the simplest case, d and d' are Euclidean distances: $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$
- *(Why is it not great?)*

Stochastic neighbor embedding

- Idea: Define similarities using probability distributions that preserve relative neighborhood relationships, rather than relying on absolute distances – which become less meaningful in high dimensions:

$$\text{For } i \neq j, \quad p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}$$

$$\text{For } i = j, \quad p_{i|i} = 0 \quad \text{such that} \quad \sum_j p_{j|i} = 1 \ \forall i$$

"The similarity of datapoint $\mathbf{x}_j$ to datapoint $\mathbf{x}_i$ is the conditional probability $p_{j|i}$, that $\mathbf{x}_i$ would pick $\mathbf{x}_j$ as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at $\mathbf{x}_i$."

— *van der Maaten and Hinton*[9]

- The Gaussian bandwidth σ_i is defined to be large in low density parts of the space, and smaller in higher density density

[9] Original paper: Van der Maaten & Hinton, Visualizing Data using t-SNE

Gaussian bandwidth σ_i

- σ_i is defined to be large in low density parts of the space, and smaller in higher density density
- It is selected such that the entropy H of the conditional distribution P_i equals a predefined value, related to the perplexity ρ

$$H_{\sigma_i}[P_i] = -\sum_j p_{j|i} \log p_{j|i}$$

$$\text{Select } \sigma_i \text{ such that } \quad \rho = 2^{H_{\sigma_i}[P_i]}$$

"Perplexity can be interpreted as a smooth measure of the effective number of neighbors."

Low dim representation

- Similarly to high dim space, define a similarity in low-dim space using a probability distribution q:

$$q_{j|i} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_{k \neq i} \exp(-\|\mathbf{z}_i - \mathbf{z}_k\|^2)}$$

- Now all that is left to do to obtain a low-dim visualization of the high-dim space is to make q as close to p as possible

Low dim representation

- Similarly to high dim space, define a similarity in low-dim space using a probability distribution q:

$$q_{j|i} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_{k \neq i} \exp(-\|\mathbf{z}_i - \mathbf{z}_k\|^2)}$$

- Now all that is left to do to obtain a low-dim visualization of the high-dim space is to make q as close to p as possible

$$\underset{\mathbf{z}_1,\ldots,\mathbf{z}_N}{\text{minimize}}\ KL[p||q] = -\sum_{\substack{i,j \\ i \neq j}} p_{j|i} \log \frac{q_{j|i}}{p_{j|i}}$$

- This is achieved using gradient descent on the coordinates of $\mathbf{z}_1, \ldots, \mathbf{z}_N$

t-distributed sochastic neighbor embedding

t-SNE introduced two variants with respect to SNE:

- Symetrized distances $p_{j|i}$ and $p_{i|j}$:

$$p_{i,j} = p_{j,i} = \frac{p_{j|i} + p_{j|i}}{2N}$$

- The use of a larger tail Student t-distribution for the probability density function in low dim:

$$q_{i,j} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

- The second point is to prevent "crowding" effects, where too many points end of clustering together in low-dim: the Student t-distribution has longer tails than Gaussians, allowing points to be more spread out

t-distributed sochastic neighbor embedding

- The overall process remains the same: perform gradient descent on $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)^\top \in \mathbb{R}^{N \times H}$[10] to minimize $KL[p||q]$

$$\mathbf{Z}_{t+1} = \mathbf{Z}_t - \alpha \frac{\partial D_{\mathsf{KL}}[p||q]}{\partial \mathbf{Z}_t}$$

- We could prove that

$$\frac{\partial KL[p||q]}{\partial \mathbf{z}_i} = 4 \sum_j (p_{ij} - q_{ij}) \frac{\mathbf{z}_i - \mathbf{z}_j}{1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2}$$

- Now you have all the elements needed to implement t-SNE yourself!

[10] With t-SNE, we almost always select $H = 2$

t-SNE illustration

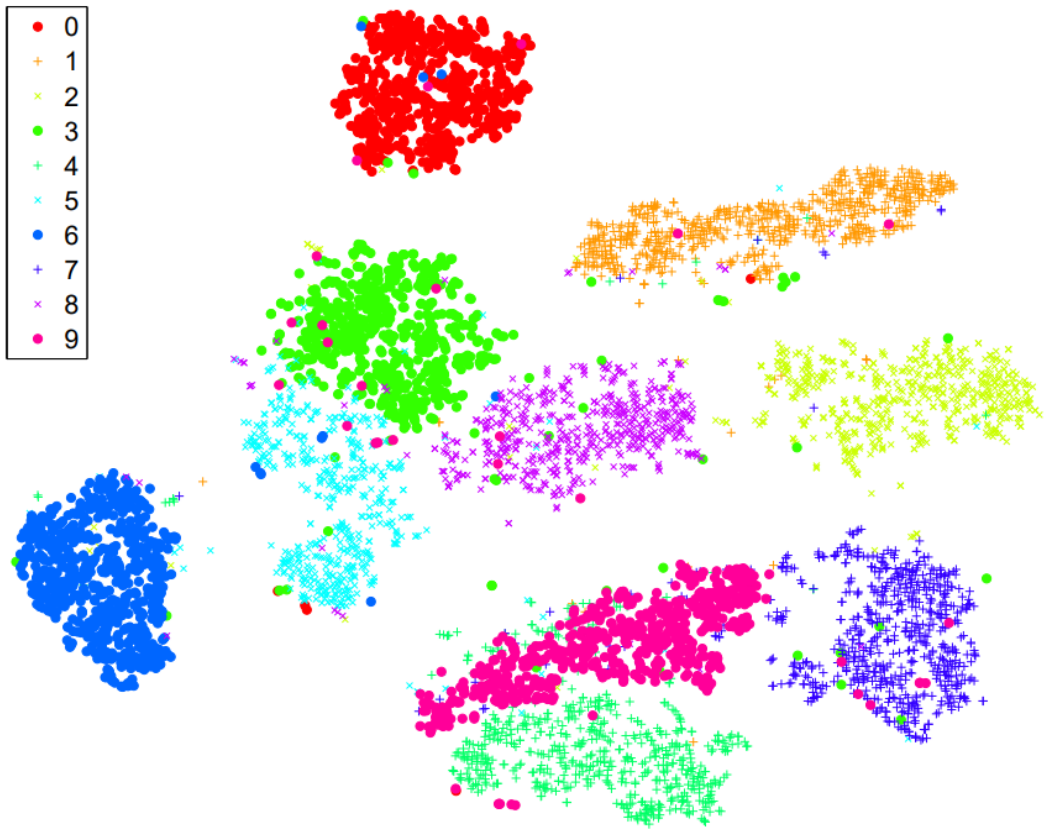


Figure: Illustration of t-SNE on the MNIST dataset[11]

[11] From Van der Maaten *et al.*, Visualizing Data using t-SNE, JMLR, 2088

MDS illustration

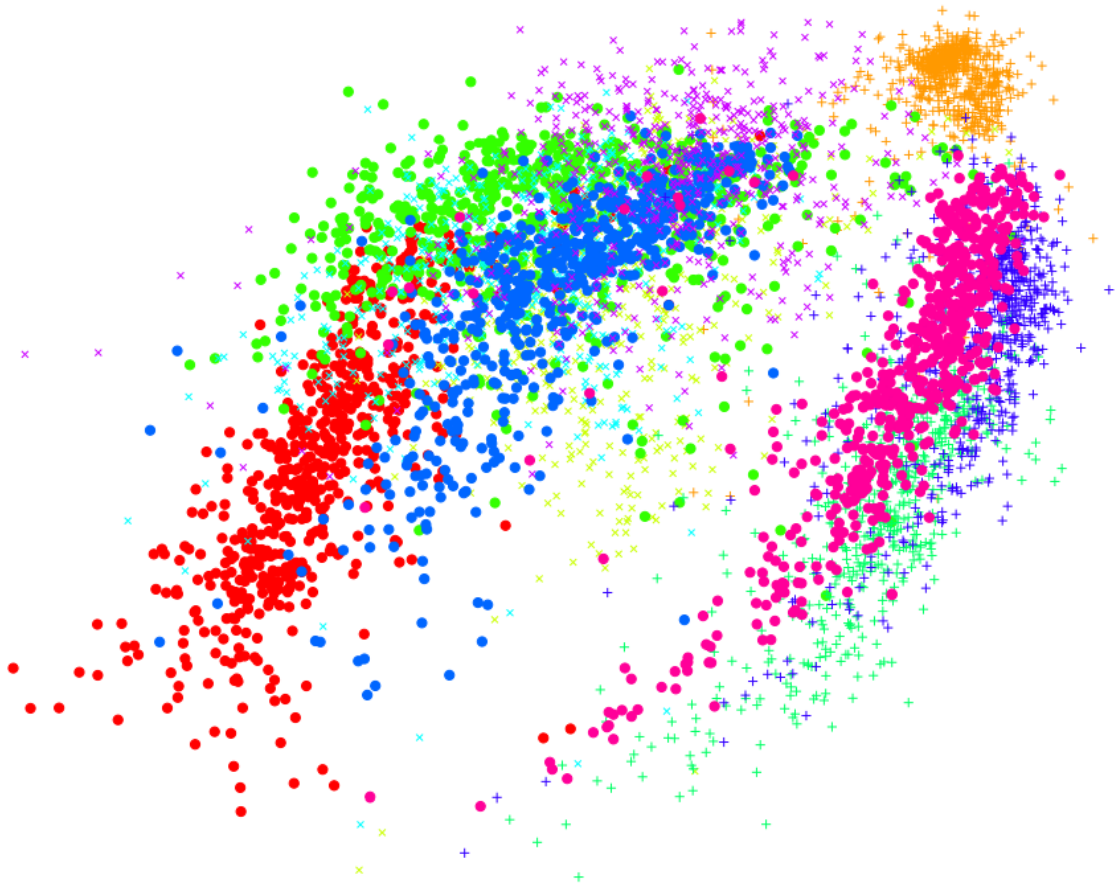


Figure: Illustration of Isomap (an extension of MDS) on the MNIST dataset[12]

[12] From Van der Maaten *et al.*, Visualizing Data using t-SNE, JMLR, 2088