Model Combination, Decision Trees, Random Forests

Milan Straka

November 30, 2020

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Covariance

Given a collection of random variables $\mathrm{x}_1, \ldots, \mathrm{x}_N$, we know that

$$\mathbb{E}\left[\sum\nolimits_i \mathrm{x}_i\right] = \sum_i \mathbb{E}\big[\mathrm{x}_i\big].$$

But how about $\mathrm{Var}(\sum_i \mathrm{x}_i)$?

$$\begin{aligned}
\mathrm{Var}\left(\sum\nolimits_i \mathrm{x}_i\right) &= \mathbb{E}\left[\left(\sum\nolimits_i \mathrm{x}_i - \sum\nolimits_i \mathbb{E}[\mathrm{x}_i]\right)^2\right] \\
&= \mathbb{E}\left[\left(\sum\nolimits_i \big(\mathrm{x}_i - \mathbb{E}[\mathrm{x}_i]\big)\right)^2\right] \\
&= \mathbb{E}\left[\sum\nolimits_i \sum\nolimits_j \big(\mathrm{x}_i - \mathbb{E}[\mathrm{x}_i]\big)\big(\mathrm{x}_j - \mathbb{E}[\mathrm{x}_j]\big)\right] \\
&= \sum_i \sum_j \mathbb{E}\Big[\big(\mathrm{x}_i - \mathbb{E}[\mathrm{x}_i]\big)\big(\mathrm{x}_j - \mathbb{E}[\mathrm{x}_j]\big)\Big].
\end{aligned}$$

Covariance

We define **covariance** of two random variables x, y as

$$\operatorname{cov}(\mathrm{x}, \mathrm{y}) = \mathbb{E}\Big[\big(\mathrm{x} - \mathbb{E}[\mathrm{x}]\big)\big(\mathrm{y} - \mathbb{E}[\mathrm{y}]\big)\Big].$$

Then,

$$\operatorname{Var}\Big(\sum\nolimits_i \mathrm{x}_i\Big) = \sum_i \sum_j \operatorname{cov}(\mathrm{x}_i, \mathrm{x}_j).$$

Note that $\operatorname{cov}(\mathrm{x}, \mathrm{x}) = \operatorname{Var}(\mathrm{x})$ and that we can write covariance as

$$\begin{aligned}\operatorname{cov}(\mathrm{x}, \mathrm{y}) &= \mathbb{E}\Big[\big(\mathrm{x} - \mathbb{E}[\mathrm{x}]\big)\big(\mathrm{y} - \mathbb{E}[\mathrm{y}]\big)\Big] \\ &= \mathbb{E}\big[\mathrm{x}\mathrm{y} - \mathrm{x}\mathbb{E}[\mathrm{y}] - \mathbb{E}[\mathrm{x}]\mathrm{y} + \mathbb{E}[\mathrm{x}]\mathbb{E}[\mathrm{y}]\big] \\ &= \mathbb{E}\big[\mathrm{x}\mathrm{y}\big] - \mathbb{E}\big[\mathrm{x}\big]\mathbb{E}\big[\mathrm{y}\big].\end{aligned}$$

Corellation

Two random variables x, y are **uncorrelated**, if $\operatorname{cov}(\mathrm{x}, \mathrm{y}) = 0$; otherwise, they are **correlated**.

Note that two *independent* random variables are uncorrelated, because

$$\begin{aligned}
\operatorname{cov}(\mathrm{x}, \mathrm{y}) &= \mathbb{E}\Big[\big(\mathrm{x} - \mathbb{E}[\mathrm{x}]\big)\big(\mathrm{y} - \mathbb{E}[\mathrm{y}]\big)\Big] \\
&= \sum_{x,y} P(x, y)\big(x - \mathbb{E}[x]\big)\big(y - \mathbb{E}[y]\big) \\
&= \sum_{x,y} P(x)\big(x - \mathbb{E}[x]\big)P(y)\big(y - \mathbb{E}[y]\big) \\
&= \sum_{x} P(x)\big(x - \mathbb{E}[x]\big)\sum_{y} P(y)\big(y - \mathbb{E}[y]\big) \\
&= \mathbb{E}_{\mathrm{x}}\big[\mathrm{x} - \mathbb{E}[\mathrm{x}]\big]\mathbb{E}_{\mathrm{y}}\big[\mathrm{y} - \mathbb{E}[\mathrm{y}]\big] = 0.
\end{aligned}$$

However, dependent random variables can be uncorrelated – random uniform x on $[-1, 1]$ and $\mathrm{y} = |\mathrm{x}|$ are not independent (y is completely determined by x), but they are uncorrelated.

Pearson correlation coefficient

There are several ways to measure correlation of random variables x, y.

Pearson correlation coefficient, denoted as ρ or r, is defined as

$$\rho \stackrel{\text{def}}{=} \frac{\operatorname{cov}(\mathrm{x}, \mathrm{y})}{\sqrt{\operatorname{Var}(\mathrm{x})}\sqrt{\operatorname{Var}(\mathrm{y})}}$$

$$r \stackrel{\text{def}}{=} \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}},$$

where:

- ρ is used when the full expectation is computed (population Pearson correlation coefficient);
- r is used when estimating the coefficient from data (sample Pearson correlation coefficient).
  - $\bar{x}$ and $\bar{y}$ are sample estimates of mean

Pearson correlation coefficient

The value of Pearson correlation coefficient is in fact normalized covariance, because its value is always bounded by $-1 \leq \rho \leq 1$ (and the same holds for r).

The bound can be derived from

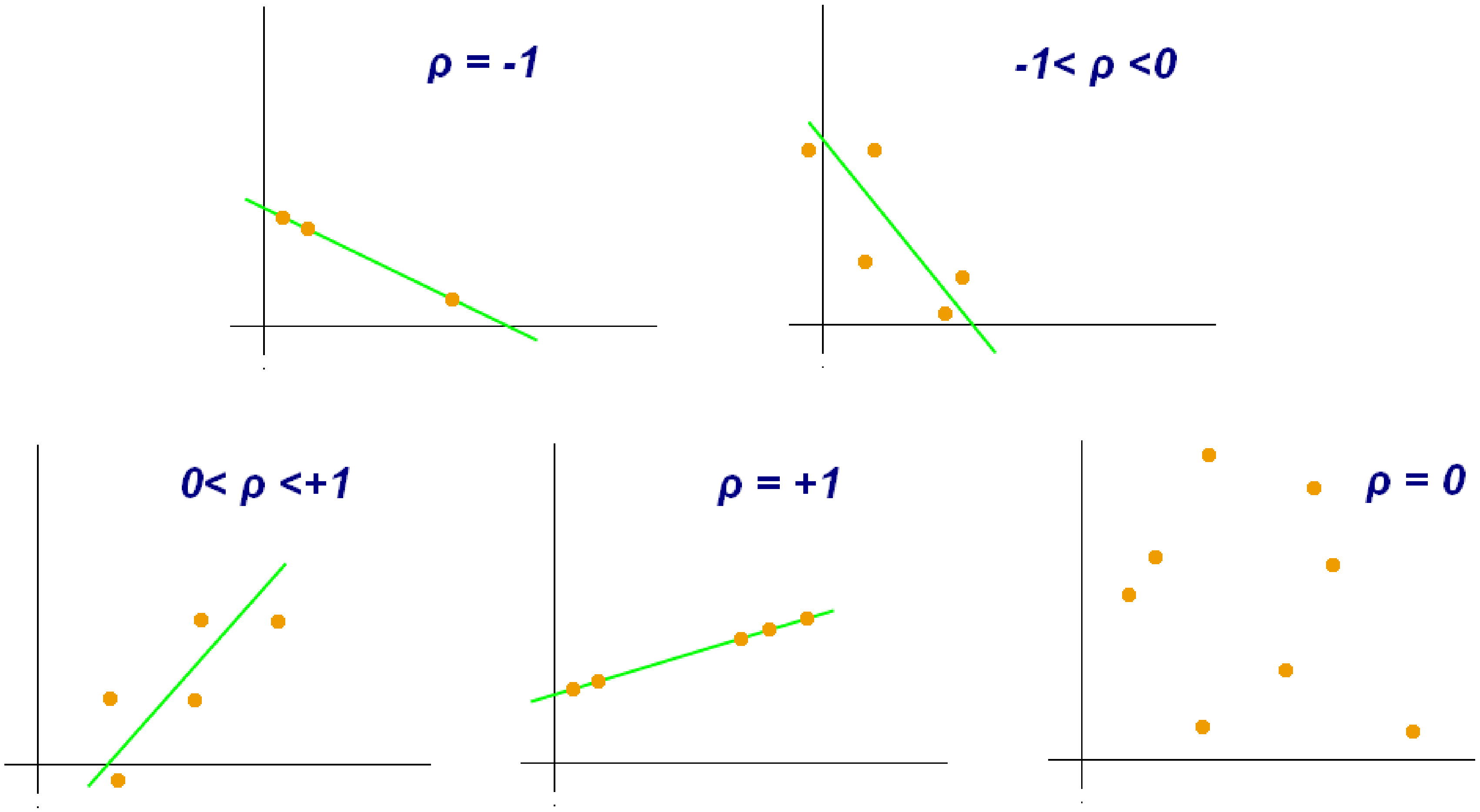
$$
\begin{aligned}
0 &\leq \mathbb{E}\left[\left(\frac{(\mathrm{x} - \mathbb{E}[\mathrm{x}])}{\sqrt{\mathrm{Var}(\mathrm{x})}} - \rho\frac{(\mathrm{y} - \mathbb{E}[\mathrm{y}])}{\sqrt{\mathrm{Var}(\mathrm{y})}}\right)^2\right] \\
&= \mathbb{E}\left[\frac{(\mathrm{x} - \mathbb{E}[\mathrm{x}])^2}{\mathrm{Var}(\mathrm{x})}\right] - 2\rho\mathbb{E}\left[\frac{(\mathrm{x} - \mathbb{E}[\mathrm{x}])}{\sqrt{\mathrm{Var}(\mathrm{x})}}\frac{(\mathrm{y} - \mathbb{E}[\mathrm{y}])}{\sqrt{\mathrm{Var}(\mathrm{y})}}\right] + \rho^2\mathbb{E}\left[\frac{(\mathrm{y} - \mathbb{E}[\mathrm{y}])^2}{\mathrm{Var}(\mathrm{y})}\right] \\
&= \frac{\mathrm{Var}(\mathrm{x})}{\mathrm{Var}(\mathrm{x})} - 2\rho \cdot \rho + \rho^2\frac{\mathrm{Var}(\mathrm{y})}{\mathrm{Var}(\mathrm{y})} = 1 - \rho^2,
\end{aligned}
$$

which yields $\rho^2 \leq 1$.

Alternatively, the desired inequality can be obtained by applying the Cauchy-Schwarz inequality $\langle u, v\rangle \leq \sqrt{\langle u, u\rangle}\sqrt{\langle v, v\rangle}$ on $\langle x, y\rangle \overset{\mathrm{def}}{=} \mathbb{E}[xy]$.

Pearson correlation coefficient

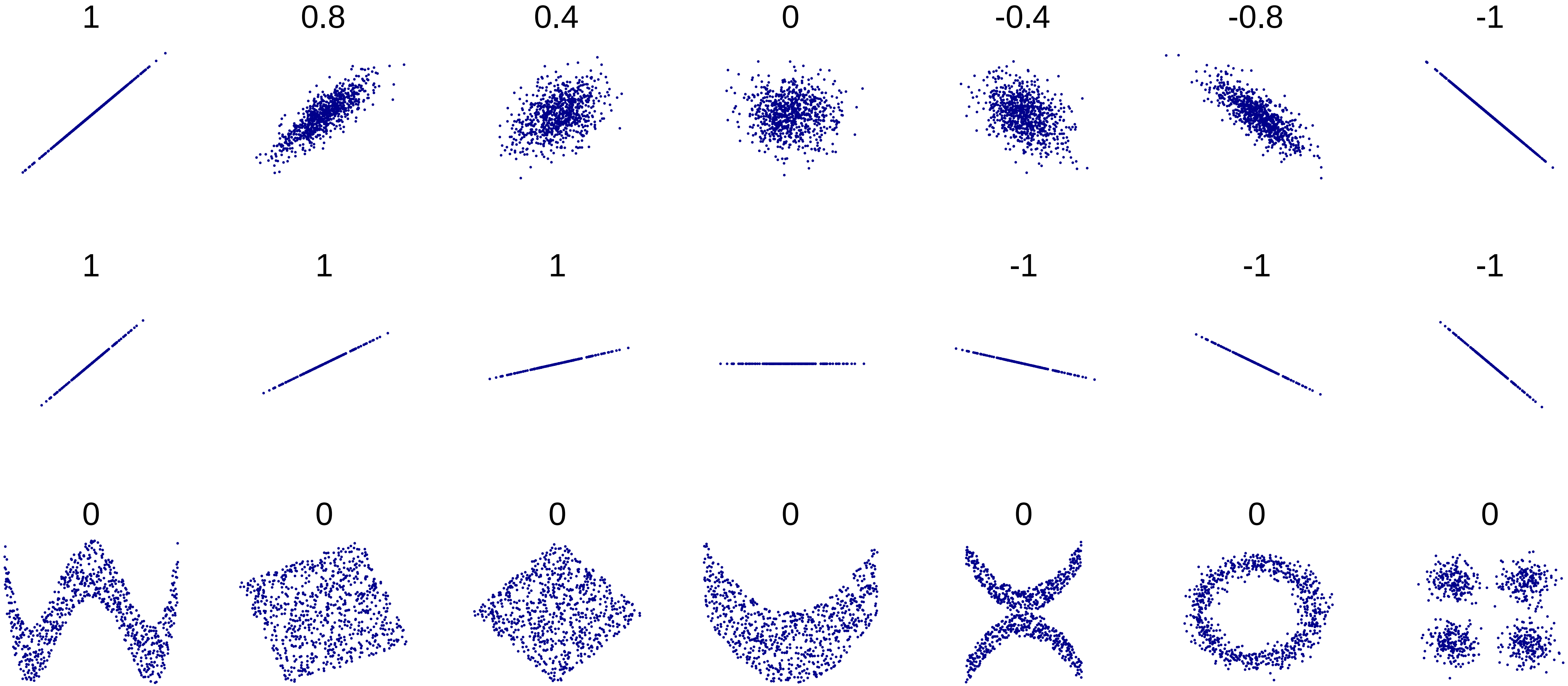
Pearson correlation coefficient quantifies **linear dependence** of the two random variables.



https://upload.wikimedia.org/wikipedia/commons/3/34/Correlation_coefficient.png

Pearson correlation coefficient

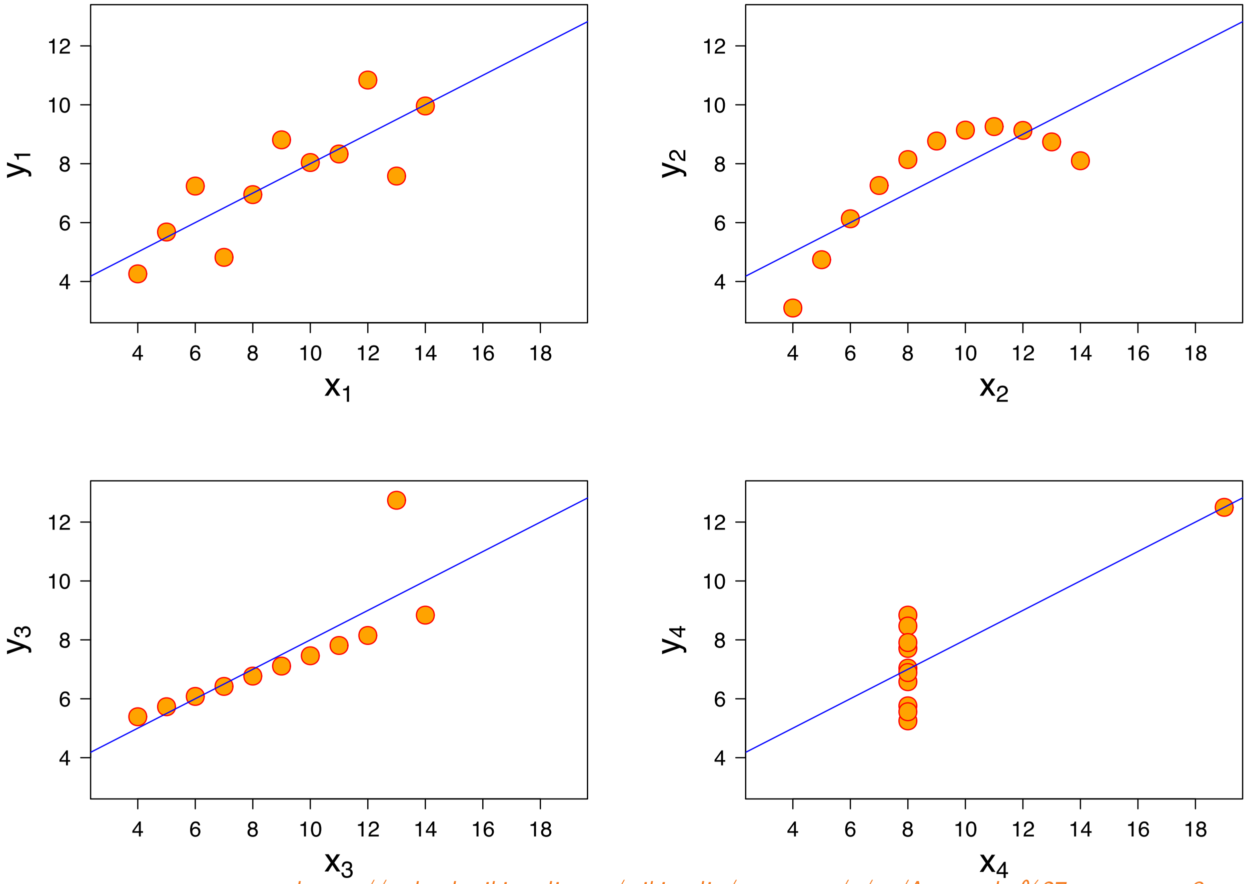
Pearson correlation coefficient quantifies **linear dependence** of the two random variables.



https://upload.wikimedia.org/wikipedia/commons/d/d4/Correlation_examples2.svg

Pearson correlation coefficient

The four displayed variables have the same mean 7.5, variance 4.12, Pearson correlation coefficient 0.816 and regression line $3 + \frac{1}{2}x$.



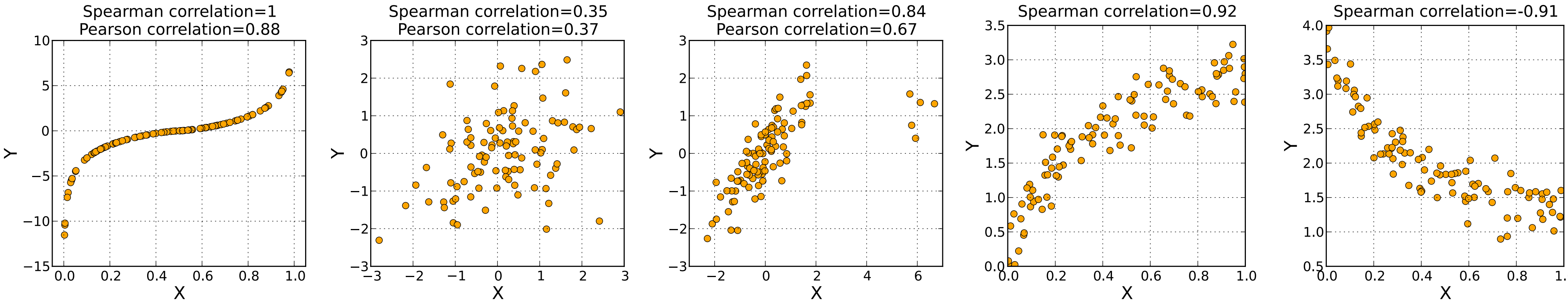
https://upload.wikimedia.org/wikipedia/commons/e/ec/Anscombe%27s_quartet_3.svg

Nonlinear Correlation – Spearman's ρ

To measure also non-linear correlation, two common coefficients are used.

Spearman's rank correlation coefficient ρ

Spearman's ρ is Pearson correlation coefficient measured on **ranks** of the original data, where a rank of an element is its index in sorted ascending order.



https://upload.wikimedia.org/wikipedia/commons/4/4e/Spearman_fig{1,2,3,5,4}.svg

Nonlinear Correlation – Kensall's τ

Kendall rank correlation coefficient τ

Kendall's τ measures the amount of *concordant pairs* (pairs where y increases/decreases when x does), minus the *discordant pairs* (where y increases/decreases when x does the opposite):

$$\begin{aligned}\tau &\stackrel{\text{def}}{=} \frac{\big|\{\text{pairs } i \neq j : x_j > x_i, y_j > y_i\}\big| - \big|\{\text{pairs } i \neq j : x_j > x_i, y_j < y_i\}\big|}{\binom{n}{2}} \\ &= \frac{\sum_{i<j} \operatorname{sign}(x_j - x_i)\operatorname{sign}(y_j - y_i)}{\binom{n}{2}}.\end{aligned}$$

There is no clear consensus whether to use Spearman's ρ or Kendall's τ, but I believe Kendall's τ is a bit more preferred. First, $\frac{1+\tau}{2}$ can be interpreted as a probability of a concordant pair, and Kendall's τ converges to a normal distribution faster.

As defined, the range of Kendall's $\tau \in [-1, 1]$. However, if there are ties, its range is smaller – therefore, several corrections (not discussed here) exist to adjust its value in case of ties.

Model Combination aka Ensembling

Ensembling is combining several models with a goal of reaching higher performance.

The simplest approach is to train several independent models and then combine their outputs by averaging or voting.

The terminology varies, but for classification:

- voting (or hard voting) usually means predicting the class predicted most often by the individual models,
- averaging (or soft voting) denotes averaging the returned model distributions and predicting the class with the highest probability.

The main idea behind ensembling is that if models have uncorrelated errors, then by averaging model outputs the errors will cancel out.

Model Combination aka Ensembling

If we denote the prediction of a model y_i on a training example $(\boldsymbol{x}, t)$ as $y_i(\boldsymbol{x}) = t + \varepsilon_i(\boldsymbol{x})$, so that $\varepsilon_i(\boldsymbol{x})$ is the model error on example $\boldsymbol{x}$, the mean square error of the model is

$$\mathbb{E}\big[(y_i(\boldsymbol{x}) - t)^2\big] = \mathbb{E}\big[\varepsilon_i^2(\boldsymbol{x})\big].$$

Considering M models, we analogously get that the mean square error of the ensemble is

$$\mathbb{E}\Bigg[\Big(\frac{1}{M}\sum\nolimits_i \varepsilon_i(\boldsymbol{x})\Big)^2\Bigg].$$

Finally, assuming that the individual errors ε_i have zero mean and are *uncorrelated*, we get that $\mathbb{E}\big[\varepsilon_i(\boldsymbol{x})\varepsilon_j(\boldsymbol{x})\big] = 0$ for $i \neq j$, and therefore,

$$\mathbb{E}\Bigg[\Big(\frac{1}{M}\sum\nolimits_i \varepsilon_i(\boldsymbol{x})\Big)^2\Bigg] = \mathbb{E}\Bigg[\frac{1}{M^2}\sum\nolimits_{i,j} \varepsilon_i(\boldsymbol{x})\varepsilon_j(\boldsymbol{x})\Bigg] = \frac{1}{M}\mathbb{E}\Bigg[\frac{1}{M}\sum\nolimits_i \varepsilon_i^2(\boldsymbol{x})\Bigg],$$

so the average error of the ensemble is $\frac{1}{M}$ times the average error of the individual models.

Bagging

For neural network models, training models with independent initialization is usually enough, given that the loss has many local minima, so the models tend to be quite independent just when using different initialization.

However, algorithms with a convex loss functions usually converge to the same optimum independent on randomization.

In these cases, we can use **bagging**, which stands for **bootstrap aggregation**.

In bagging, we construct a different dataset for every model to be trained. We construct it using **bootstrapping** – we sample as many training instances as the original dataset has, but **with replacement**.

Such dataset is sampled using the same empirical data distribution and has the same size, but is not identical.

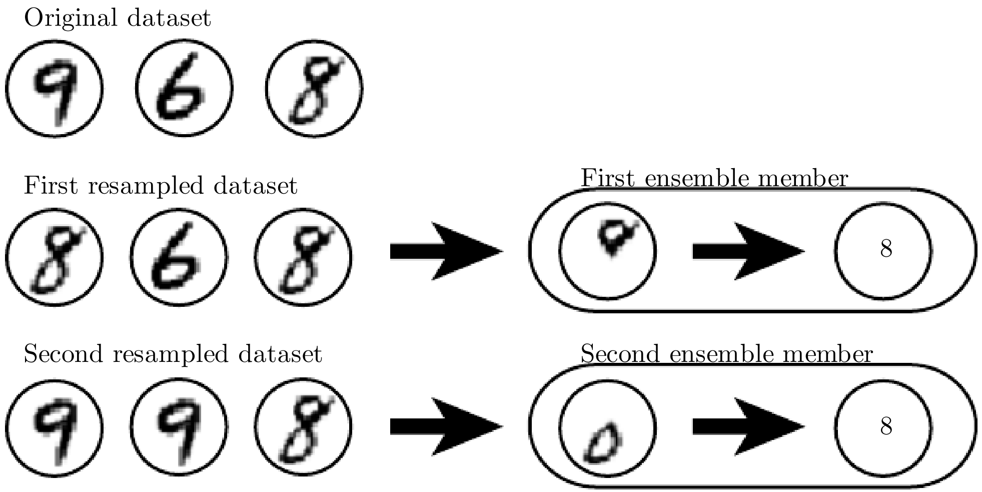


Figure 7.5, page 257 of Deep Learning Book, http://deeplearningbook.org

Decision Trees

The idea of decision trees is to partition the input space into usually cuboid regions and solving each region with a simpler model.

We focus on **Classification and Regression Trees** (CART; Breiman et al., 1984), but there are additional variants like ID3, C4.5, …

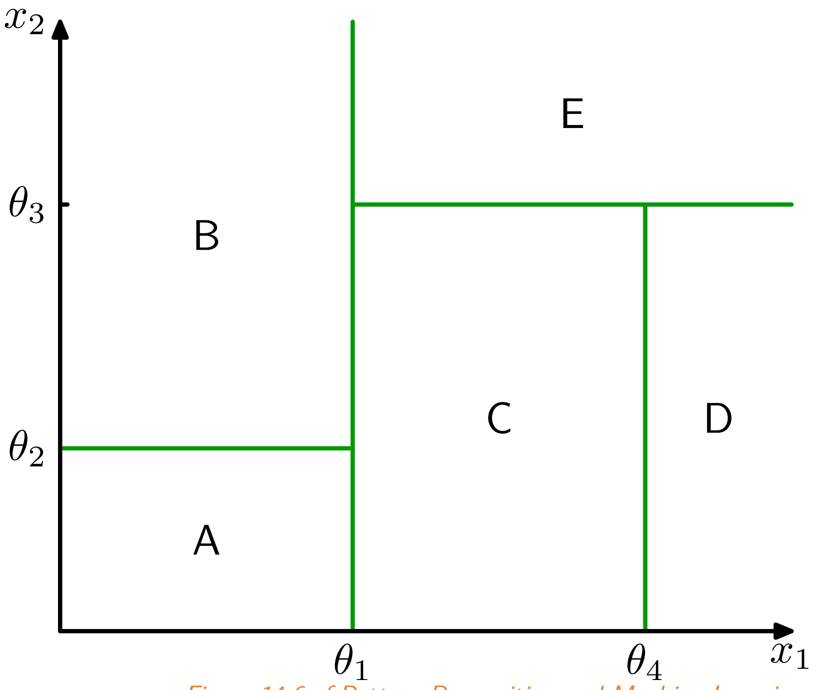


Figure 14.6 of Pattern Recognition and Machine Learning.

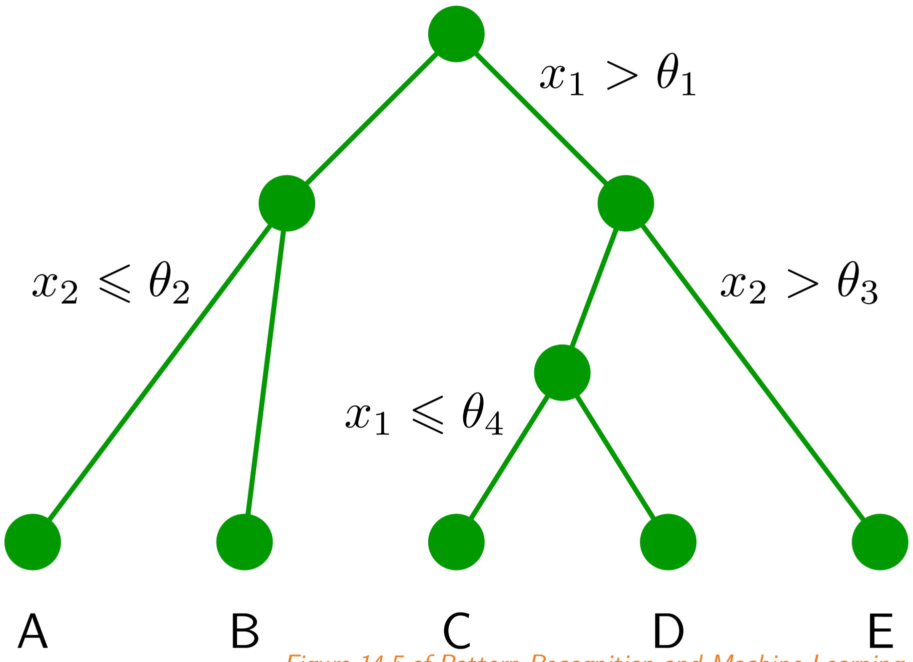


Figure 14.5 of Pattern Recognition and Machine Learning.

Regression Decision Trees

Assume we have an input dataset $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, $\boldsymbol{t} \in \mathbb{R}^N$. At the beginning, the decision tree is just a single node and all input examples belong to this node. We denote $I_\mathcal{T}$ the set of training example indices belonging to a leaf node $\mathcal{T}$.

For each leaf, our model will predict the average of the training examples belonging to that leaf, $\hat{t}_\mathcal{T} = \frac{1}{|I_\mathcal{T}|} \sum_{i \in I_\mathcal{T}} t_i$.

We will use a **criterion** $c_\mathcal{T}$ telling us how *uniform* or *homogeneous* are the training examples belonging to a leaf node $\mathcal{T}$ – for regression, we will employ the sum of squares error between the examples belonging to the node and the predicted value in that node; this is proportional to the variance of the training examples belonging to the leaf node $\mathcal{T}$, multiplied by the number of the examples. Note that even if it is not *mean* squared error, it is sometimes denoted as MSE.

$$c_\textrm{SE}(\mathcal{T}) \stackrel{\textrm{def}}{=} \sum_{i \in I_\mathcal{T}} (t_i - \hat{t}_\mathcal{T})^2, \textrm{ where } \hat{t}_\mathcal{T} = \frac{1}{|I_\mathcal{T}|} \sum_{i \in I_\mathcal{T}} t_i.$$

Tree Construction

To split a node, the goal is to find a feature and its value such that when splitting a node $\mathcal{T}$ into $\mathcal{T}_L$ and $\mathcal{T}_R$, the resulting regions decrease the overall criterion value the most, i.e., the difference $c_{\mathcal{T}_L} + c_{\mathcal{T}_R} - c_{\mathcal{T}}$ is the lowest.

Usually we have several constraints, we mention on the most common ones:

- **maximum tree depth**: we do not split nodes with this depth;
- **minimum examples to split**: we only split nodes with this many training examples;
- **maximum number of leaf nodes**: we split until we reach the given number of leaves.

The tree is usually built in one of two ways:

- if the number of leaf nodes is unlimited, we usually build the tree in a depth-first manner, recursively splitting every leaf until some of the above constraint is invalidated;
- if the maximum number of leaf nodes is given, we usually split such leaf $\mathcal{T}$ where the criterion difference $c_{\mathcal{T}_L} + c_{\mathcal{T}_R} - c_{\mathcal{T}}$ is the lowest.

Pruning

To control overfitting, the mentioned constraints can be used.

Additionally, **pruning** can also be used. After training, we might decide that some subtrees are not necessary and *prune* them (replacing them by a leaf). Pruning can be used both as a regularization or model compression.

There are many heuristics to prune a decision tree; Scikit-learn implements **minimal cost-complexity pruning**:

- we extend the criterion to *cost-complexity criterion* as
  - for a leaf, $c_\alpha(\mathcal{T}) = c(\mathcal{T}) + \alpha$,
  - for a subtree T_t with a root t, $c_\alpha(T_t) = \sum_\textrm{leaves} c_\alpha(\mathcal{T}) = \sum_\textrm{leaves} c(\mathcal{T}) + \alpha|\textrm{leaves}|$;
- generally a criterion in a node t is greater or equal to the sum of criteria of its leaves;
- $\alpha_\textrm{eff}$ is the value of α such that the above two cost-complexity quantities are equal
  - $\alpha_\textrm{eff} = \big(c(\mathcal{T}) - c(T_t)\big)/\big(|\textrm{leaves}| - 1\big)$;
- we then prune the nodes in the order of increasing $\alpha_\textrm{eff}$.

Classification Decision Trees

For multi-class classification, we predict such class most frequent in the training examples belonging to a leaf $\mathcal{T}$.

To define the criterions, let us denote the average probability for class k in a region $\mathcal{T}$ as $p_{\mathcal{T}}(k)$.

For classification trees, one of the following two criterions is usually used:

- **Gini index**:

$$c_{\textrm{Gini}}(\mathcal{T}) \stackrel{\textrm{def}}{=} |I_{\mathcal{T}}| \sum_k p_{\mathcal{T}}(k)\big(1 - p_{\mathcal{T}}(k)\big)$$

- **Entropy Criterion**

$$c_{\textrm{entropy}}(\mathcal{T}) \stackrel{\textrm{def}}{=} |I_{\mathcal{T}}| H(p_{\mathcal{T}}) = -|I_{\mathcal{T}}| \sum_{\substack{k \\ p_{\mathcal{T}}(k) \neq 0}} p_{\mathcal{T}}(k) \log p_{\mathcal{T}}(k)$$

Binary Gini as (M)SE Loss

Recall that $I_\mathcal{T}$ denotes the set of training example indices belonging to a leaf node $\mathcal{T}$, let $n_\mathcal{T}(0)$ be the number of examples with target value 0, $n_\mathcal{T}(1)$ be the number of examples with target value 1, and let $p_\mathcal{T} = \frac{1}{|I_\mathcal{T}|}\sum_{i \in I_\mathcal{T}} t_i$.

Consider sum of squares loss $\mathcal{L}(p) = \sum_{i \in I_\mathcal{T}} (p - t_i)^2$.

By setting the derivative of the loss to zero, we get that the p minimizing the loss fulfils $|I_\mathcal{T}| p = \sum_{i \in I_\mathcal{T}} t_i$, i.e., $p = p_\mathcal{T}$.

The value of the loss is then

$$\begin{aligned}
\mathcal{L}(p_\mathcal{T}) &= \sum_{i \in I_\mathcal{T}} (p_\mathcal{T} - t_i)^2 = n_\mathcal{T}(0)(p_\mathcal{T} - 0)^2 + n_\mathcal{T}(1)(p_\mathcal{T} - 1)^2 \\
&= \frac{n_\mathcal{T}(0) n_\mathcal{T}(1)^2}{\big(n_\mathcal{T}(0) + n_\mathcal{T}(1)\big)^2} + \frac{n_\mathcal{T}(1) n_\mathcal{T}(0)^2}{\big(n_\mathcal{T}(0) + n_\mathcal{T}(1)\big)^2} = \frac{\big(n_\mathcal{T}(1) + n_\mathcal{T}(0)\big) n_\mathcal{T}(0) n_\mathcal{T}(1)}{\big(n_\mathcal{T}(0) + n_\mathcal{T}(1)\big)^2} \\
&= \big(n_\mathcal{T}(0) + n_\mathcal{T}(1)\big)(1 - p_\mathcal{T}) p_\mathcal{T} = |I_\mathcal{T}| p_\mathcal{T} (1 - p_\mathcal{T})
\end{aligned}$$

Entropy as NLL Loss

Again let $I_\mathcal{T}$ denote the set of training example indices belonging to a leaf node $\mathcal{T}$, let $n_\mathcal{T}(c)$ be the number of examples with target value c, and let $p_\mathcal{T}(c) = \frac{n_\mathcal{T}(c)}{|I_\mathcal{T}|} = \frac{1}{|I_\mathcal{T}|} \sum_{i \in I_\mathcal{T}} [t_i = c]$.

Consider a distribution $\boldsymbol p$ on K classes and non-averaged NLL loss $\mathcal{L}(\boldsymbol p) = \sum_{i \in I_\mathcal{T}} -\log p_{t_i}$.

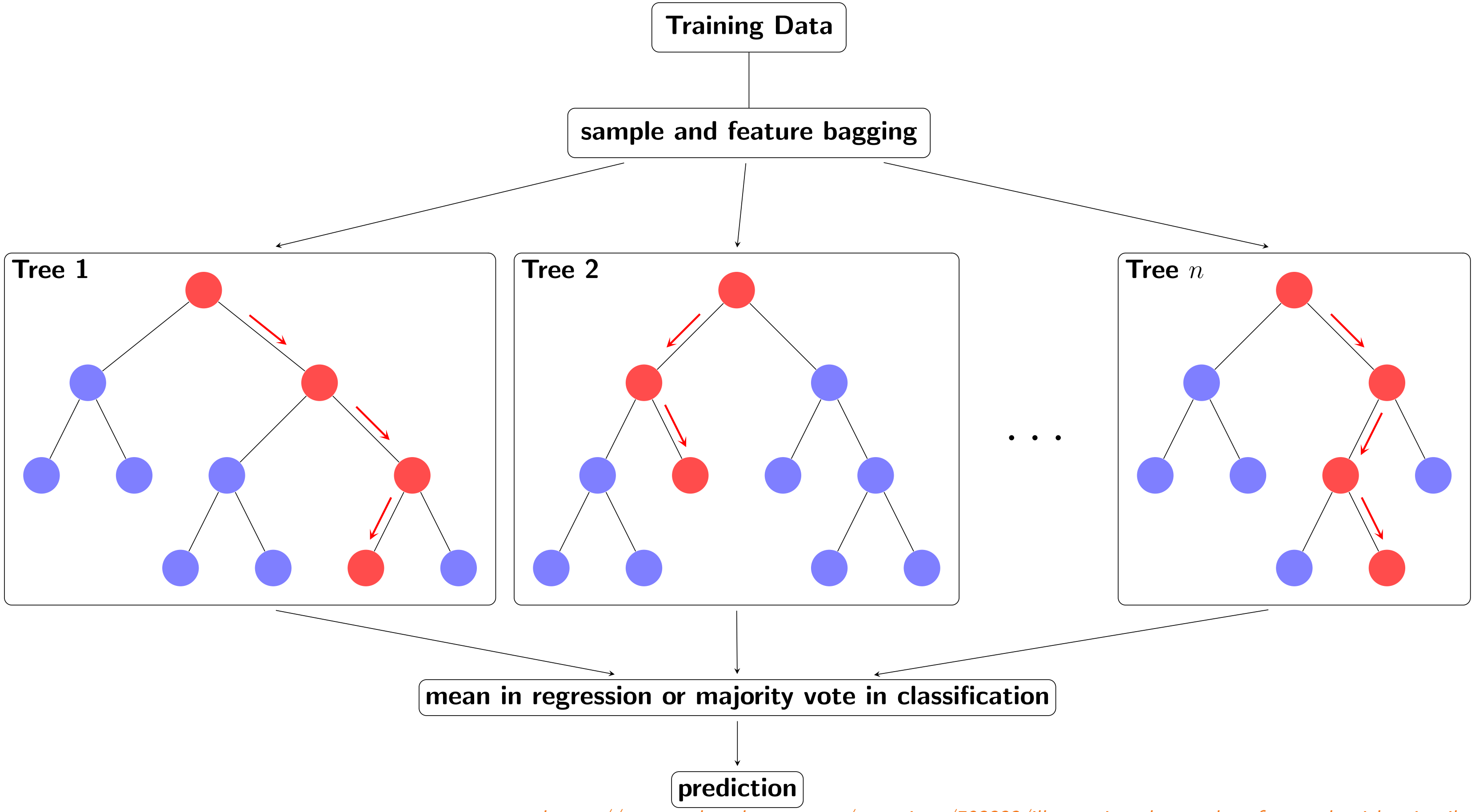
By setting the derivative of the loss with respect to p_c to zero (using a Lagrangian with constraint $\sum_c p_c = 1$), we get that the $\boldsymbol p$ minimizing the loss fulfils $p_c = p_\mathcal{T}(c)$.

The value of the loss with respect to $p_\mathcal{T}$ is then

$$\begin{aligned}
\mathcal{L}(p_\mathcal{T}) &= \sum_{i \in I_\mathcal{T}} -\log p_{t_i} \\
&= - \sum_{\substack{c \\ p_\mathcal{T}(c) \neq 0}} n_\mathcal{T}(c) \log p_\mathcal{T}(c) \\
&= -|I_\mathcal{T}| \sum_{\substack{c \\ p_\mathcal{T}(c) \neq 0}} p_\mathcal{T}(c) \log p_\mathcal{T}(c) = |I_\mathcal{T}| H(p_\mathcal{T}).
\end{aligned}$$

Random Forests

Bagging of data combined with random subset of features (sometimes called *feature bagging*).



Training Data
sample and feature bagging
Tree 1
Tree 2
. . .
Tree n
mean in regression or majority vote in classification
prediction

https://tex.stackexchange.com/questions/503883/illustrating-the-random-forest-algorithm-in-tikz

Random Forests

Bagging

Every decision tree is trained using bagging (on a bootstrapped dataset).

Random Subset of Features

During each node split, only a random subset of features is considered, when finding the best split. A fresh random subset is used for every node.

Extra Trees

The so-called extra trees are even more randomized, not finding the best possible feature value when choosing a split, but considering uniformly random samples from a feature's empirical range (minimum and maximum in the training data).

Demos

https://cs.stanford.edu/~karpathy/svmjs/demo/demoforest.html