Lecture 5 - 07-04-2020

1.1 Tree Classifier

Supposed we groped a tree up to this point and we are wandering how to grow it.
S Training set $(x_1, y_1)...(x_m, y_m)$, $x_1 \in X$

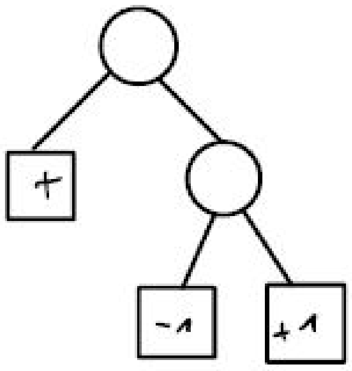


Figure 1.1: Example of domain of K_{NN}

$$S_\ell \equiv \{(x_1, y_1)\, x_t \quad \textit{is router to } \ell\}$$

$y_1 \in \{-1, 1\}$

$$S_\ell^+ \equiv \{(x_1, y_1) \in S_\ell : \quad y_t = +1\}$$
$$S_\ell^- \equiv \{(x_1, y_1) \in S_\ell : \quad y_t = -1\} \qquad S_\ell^+ \cap S_\ell^- \equiv 0 \qquad S_\ell \equiv S_\ell^+ \cup S_\ell^-$$

$$N_\ell = |S_\ell| \qquad N_\ell^+ = |S_\ell^+| \qquad N_\ell^- = |S_\ell^-|$$
$$N_\ell = N_\ell^- + N_\ell^+$$

leaf ℓ classifies all traning example (S_ℓ)

$$Y_\ell = \begin{cases} +1, & \text{If } N_\ell^+ \geq N_\ell^- \\ -1, & \text{If } \textit{otherwise} \end{cases}$$

ℓ makes a mistake on $min\{N_\ell^+, N_\ell^-\}$ example in S_ℓ

$$\hat{\ell}(h_T) = \frac{1}{m} \cdot \sum_\ell min\{\frac{N_\ell^+}{N_\ell}, \frac{N_\ell^-}{N_\ell}\} \cdot N_\ell =$$

$$= \frac{1}{m} \cdot \sum_\ell \psi \cdot (\frac{N_\ell +}{N_\ell}) \cdot N_\ell \quad \longrightarrow \quad \frac{N_\ell^+}{N_\ell} = 1 - \frac{N_\ell}{N_\ell ??}$$

where $\psi(a) = min\{a, 1-a\} \qquad a \in [0,1]$
I want to replace inner node with other leaves.

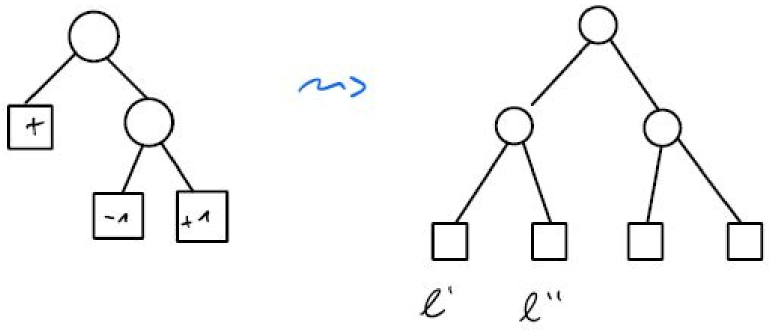


Figure 1.2: Example of domain of K_{NN}

How is traning error going to change? (when i replace inner nodes with other leaves)
I'm hoping my algorithm is not going to overfit (if training error goes to 0 also testing error goes to 0).

1.2 Jensen's inequality

If ψ is a concave function $\longrightarrow$ (like *log* or $\sqrt[2]{..}$)

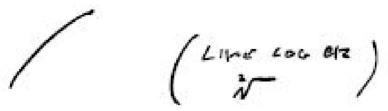


Figure 1.3: Example of domain of K_{NN}

Also ψ is a function that map 0 to 1, $\quad\longrightarrow\quad \psi\ [0,1] \to \mathbb{R}$

$\psi(\alpha \cdot a + (1-\alpha)\cdot b) \geq \alpha \cdot \psi(a) + (1-\alpha)\cdot\psi(b)$ *Also 2° derivative is negative*

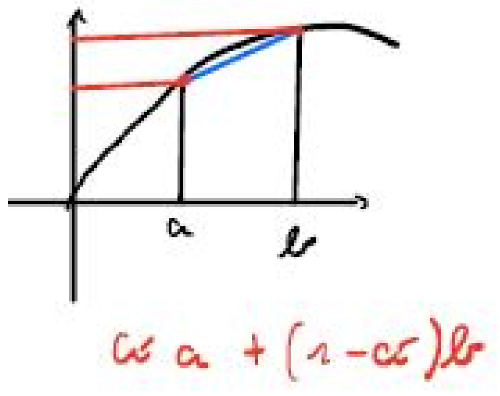


Figure 1.4: Example of domain of K_{NN}

$$\hat{\ell}(h_T) = \frac{1}{m} \cdot \sum_{\ell} \psi(\frac{N_\ell^+}{N_\ell}) \cdot N_\ell$$

Look a single contribution fo a leaf ℓ to training error

$$\psi(\frac{N_\ell^+}{N_\ell}) \cdot N_\ell = \psi(\frac{N_\ell'^+}{N_\ell'} \cdot \frac{N_\ell'}{N_\ell} + \frac{N_\ell"^+}{N_\ell"} \cdot \frac{N_\ell"}{N_\ell}) \cdot N_\ell$$

where $\frac{N_\ell'}{N_\ell} = \alpha$ and $\frac{N_\ell"}{N_\ell} = 1-\alpha$ so $\frac{N_\ell'}{N_\ell} + \frac{N_\ell"}{N_\ell} = 1$ $\longrightarrow \alpha + 1 - \alpha = 1$

$N_{\ell'}^{+} + N_{\ell"}^{+} = N_{\ell}$

I want to check function min concave between 0 and 1.

$$min(0,1) = 0 \qquad \psi(a) = min(\alpha, 1-\alpha)$$

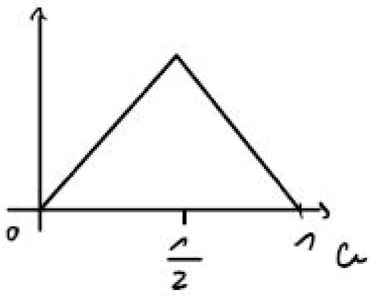


Figure 1.5: Example of domain of K_{NN}

This is a concave function and now I can apply Jensen's inquality

$$\psi(\frac{N_{\ell}^{+}}{N_{\ell}}) \cdot N_{\ell} \geq (\frac{N_{\ell}'}{N_{\ell}} \cdot \psi(\frac{N_{\ell}'^{+}}{N_{\ell}'}) + \frac{N_{\ell}"}{N_{\ell}} \cdot \psi(\frac{N_{\ell}"^{+}}{N_{\ell}"})) \cdot N_{\ell} =$$

$$= \boxed{\psi(\tfrac{N_{\ell}'^{+}}{N_{\ell}'}) \cdot N_{\ell}' + \psi(\tfrac{N_{\ell}"^{+}}{N_{\ell}"}) \cdot N_{\ell}"}$$

This are the contribuion of ℓ' and $\ell"$ to the training error

Every time i split my tree my training error is never going to increase since we have a concave function.
Whenever I'm growing my tree training error is going to be smaller.

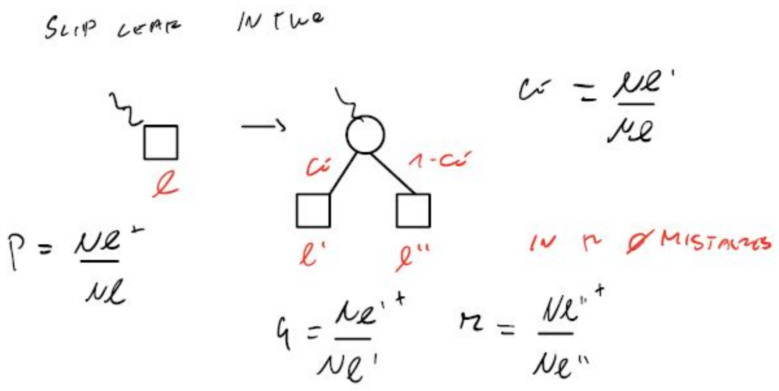
Every time a leaf is expanded the training error never goes up. (Hopelly will go down)
I'll should always grow the tree by expanding leave that decrease the training error as much as possible.
If i take the effort of growing the tree i should get benefits. I can imaging that if i grow the tree at random my training error is going to drop down error (but maybe will derive overfitting).
For now is just an intuition since we will introduced statistical learning model.

Could be complicated and tree big may have 100 leave and there could be many way of associating a test with that leaves.
I can spent a lot of time to select which leave is the best promising to split.

- Grow the tree by expanding leave that decrease the training error as much as possible
- In general we can assume:
  greedy algorithm at each step pick the pair leaf and test that cause (approximative) the largest decrease in training error

In practise we want optimise this all the way since it's time expensive. That's the approximately since we are not every time sure.

— MANCA PARTE —



SLIP LEAFE IN TWO
ℓ
c_i
$1-c_i$
ℓ'
ℓ''
$c_i = \frac{N_{\ell'}}{N_\ell}$
$p = \frac{N_\ell^+}{N_\ell}$
IN r ∅ MISTAKES
$q = \frac{N_{\ell'}^+}{N_{\ell'}}$
$r = \frac{N_{\ell''}^+}{N_{\ell''}}$

Figure 1.6: Example of domain of K_{NN}

$p = 0.8 \qquad q = 1 \qquad r = 1 \qquad \alpha = 60\%$
Net Change in number of mistakes

$$\psi(p) - (\alpha \cdot \psi(q) + (1 - \alpha) \cdot \psi(r)) =$$

$$\ell \quad - \quad \ell' \quad + \quad \ell''$$

Fraction of example miss classified $\ell-$ error $\ell'+$ error ℓ''

$$= 0.2 - (\frac{1}{2} \cdot 0.4 + \frac{1}{2} \cdot 0) = 0$$

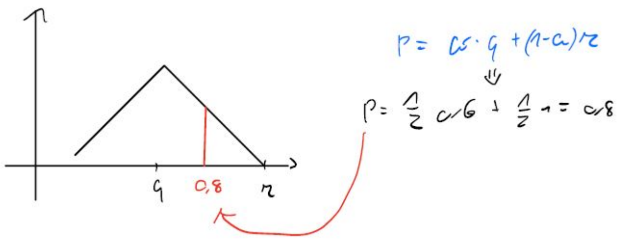


Figure 1.7: Example of domain of K_{NN}

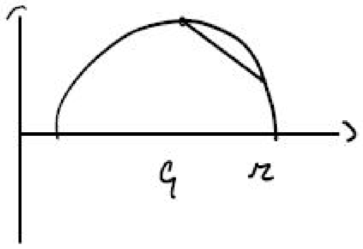


Figure 1.8: Example of domain of K_{NN}

Idea is to replace minimum function with convex combination.

$$\psi(\alpha) = min\ \{\alpha, 1-\alpha\} \qquad \psi(a) \geq \psi(\alpha)$$

$$\begin{cases} \psi_1(\alpha) = 2 \cdot \alpha \cdot (1-\alpha) \longrightarrow GNI \\ \psi_2(\alpha) = -\frac{\alpha}{2} \cdot \ln\alpha - \frac{1-\alpha}{2} \cdot \ln(1-\alpha) \longrightarrow ENTROPY \\ \psi_3(\alpha) = \sqrt{\alpha \cdot (1-\alpha)} \end{cases}$$

All this functions has this shape (concave???)

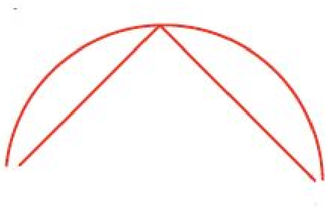


Figure 1.9: Example of domain of K_{NN}

In practise Machine Learning algorithm use GNI or entropy to control the split

1.3 Tree Predictor

- Multi class classification $|Y| > 2 \longrightarrow$ take majority
- Regression $Y = \mathbb{R} \longrightarrow$ take average of labels in S_ℓ

I still take majority among different classes.
Take average of labels in S_ℓ
Unless $\frac{N_\ell^+}{N_\ell} \in 0, 1 \qquad \forall$ leaves ℓ, $\hat{\ell}(h_T) > 0$
Unless leaves are *"pured"*, the training error will be bigger than 0.

In general, i can always write $\hat{\ell}(h_t)$ to 0 by growing enough the tree unless there are x_1 in the Time Series such that $(x_t, y_t)(x_t, y_t')$ with $y_t \neq y_t'$ both occur.

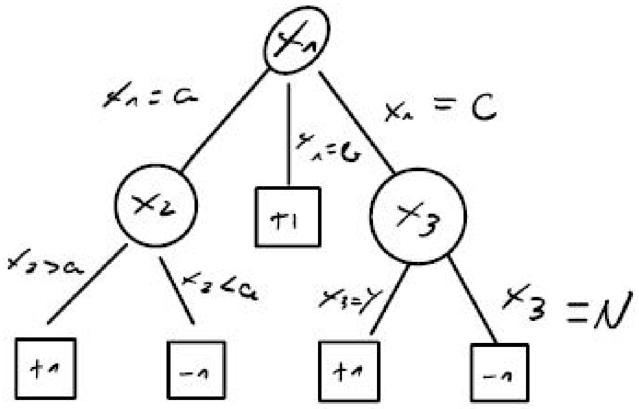


Figure 1.10: Example of domain of K_{NN}

$$if (x_1 = \alpha) \wedge (x_2 =\geq \alpha) \vee (x_1 = b) \vee (x_1 = c) \wedge (x_3 = y)$$

then predict 1

else

then predict -1

— Picture of tree classifier of iris dataset. —
I'm using due attribute at the time.
Each data point is a flower and i can measure how petal and sepal are long.

I can use two attribute and i test this two. I can see the plot of the tree classifier (second one) making test splitting data space into region that has this sort of “blackish” shape (like boxes: blue box, red box, yellow box)
A good exercise in which I want to reconstruct the tree given this picture.

1.4 Statistical model for Machine Learning

To understand Tree classifier, nearest neighbour and other algorithm...
It’s important to understand that the only way to have a guideline in which model to choose.

This mathematical model are developed to learning and choose learning algorithm.

Now let start with theoretical model.

- How example (x, y) are generated to create test set and training set?
  We get the dataset but we need to have a mathematical model for this process. (x, y) are drawn from a fixed but unknown probability distribution on the pairs X and Y (X data space, Y label set o label space)

- Why X should be random?
  In general we assumed that not all the x in X are equally likely to be observed. I have some distribution over my data point and this said that I’m most like to get a datapoint to another.

- How much label?
  Often labels are not determined uniquely by their datapoints because labels are given by human that have their subjective thoughts and also natural phenomena. Labels are stochastic phenomena given a datapoint: i will have a distribution.

We’re going to write (in capital) (X, Y) since they are random variable drawn from D on $X \cdot Y$ A dataset $(X_1, Y_1)...(X_m, Y_m)$ they are drawn independently from D (distribution on examples)
When I get a training the abstraction of process collecting a training set
D is a joint probability distribution over $X \cdot Y$
where D_x is the marginal over $X \rightarrow D_y|x$ (conditional of Y given X).

I can divided my draw in two part. I draw sample and label from conditional.??
Any dataset (training or test) is a random sample (campione casuale) in the statistical sense $\longrightarrow$ so we can use all stastical tools to make inference.