Supervised Learning in Python

- Classification Algorithms in this course:
  - ↳ KNN, N. Bayes, Decision Tree, Perceptron
  - ↳ Pros vs Cons
- Practical topics:
  hyperparams, CV, feature extraction & selection

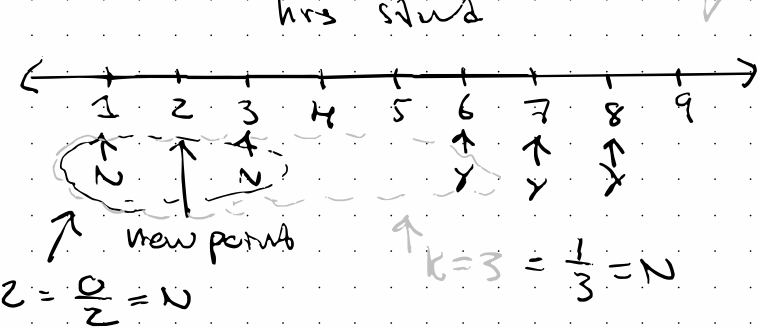
→ k-nearest neighbors

example: will you pass an exam?

↳ find students who studied for same time & estimate outcome based on them.

student	hrs studied	pass
		1
		0
		1

based on distance.



→ Prediction pseudocode

1 nearest neighbour.

```
predict(x0)
    closest-dist = inf
    "    -class = -1
    for x, y in train-dat
        d = dist(x, x0)
        if d < closest-dist
            closest-dist = d,
            closest-class = y
    return closest class
```

↳ Notice how calculation gets more brutal w/ large datasets.

↳ Optimize performance w/ sorting algorithms

Break ties?
↳ argmax or pick at random

How to choose k?
↳ Cross validation

→ When KNN can fail

↳ data, grid of alternating dots

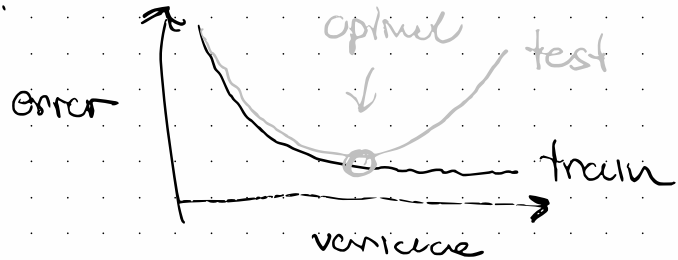
Always will classify the other class (due to alternation)

→ Effect of K

↳ Small values of k create smooth boundaries. (high bias)

↳ Large vals give flexible boundaries (high var)

Need to balance the two so you don't overfit.



→ knn pseudocode using matrix algebra

X has shape $m_x \times n_x$ (100×2)

$\hat{y}$ has shape $m_y \times n_y$ (100×1)

$\hat{X}_{new} \in (1, 2)$

↖ as many examples as wanted

$X_{new2} \in \hat{X}_{new}.\text{repeat}(100, 2)$

$D \; \hat{d} = X^2 - X_{new}$

$\hat{d} = \hat{d}$ add along rows

↳ argmin of $\hat{d}$ gives closest point

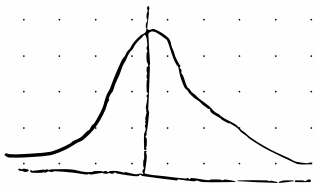
↳ most frequent of remaining vals gives classification

→ Naive Bayes

↳ from conditional probability

$P(A|B)$ ← prob. A given B

ie. given a person's height, what is prob of m/f



→ Probabilities (continuous)

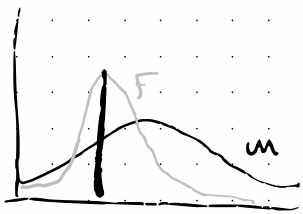
↳ $f(x) = pdf(x)$ ← not a probability but prob. DENSITY

$P(g) \sim B(0.5)$ $\quad g = \begin{cases} 0: \text{male} \\ 1: \text{female?} \end{cases}$

$P(h|g) \sim N(70in, 16in^2) I_{g=0}$

$+ N(65in, 12.5in^2) I_{g=1}$

↳ given our assumption of the probability of each gender & the data over heights we can predict somewhat gender given their height



↑ more likely to be female

↳ note that normalisation is not always necessary
↳ just choose argmax for classification
↳ comparing between the same $P(x)$?

Also at high dimensions prior & likelihoods start to reach 0.
↳ "space" between points gets larger

$k^* = \text{argmax}_k (\log P(x|y) + \log P(y))$

↑ use log to make calculations robust to small probabilities (esp w/ high dim.)

→ Probabilities (Discrete)

↳ example: spam classification

email → cleaning wrangling → feature extraction
↳ # word appearance in corpus

↳ word is same or not
↳ series of Bernauli trials ⇒ Binomial

if $n = 100$ & θ = prob of specific word = 8%

& it appears 5 times (k)

$$f(5) = \binom{100}{5} 0.08^5 (1 - 0.08)^{100-5} = 9\%$$

ie we picked 100 words & got 5 of interest w/ $p(\theta) = .08$

↳ there are 100c5 ways we could have done this

↳ thus we can calculate which is more likely θ { spam = 8%
{ ¬spam .1%

↳ get $p(X|\theta)p(\theta)$ for all θ
& compare w/o norm w/ $p(X)$

↳ increase the number of features w/ multinomial

(lazy programmer kind of sucks at teaching this)

Naive Bayes

↳ sometimes features are not independent

independence is assumed!

Pseudocode

```
def fit(X, y)
    gauss-dict = dict
    priors = dict
    for c in classes:
        Xc = X[corresponding y = c]
        mu, var = mean & diagonal cover of Xc
        dict of gauss(c) mu, var
        priors[c] = len(Xc)/len(X)
```

```
def predict(X):
    predictions = []
    max_post = -inf
    best_class = none
    for x in X:
        for c in classes:
        mu, var = dict of gaussians[c]
        post = log_pdf(x, mu, var)
                + log(priors[c])
        if post > max_post:
            max_post = post
            best_class = c
        pred.append best_class
    return pred
```

Non-Naive Bayes

↳ not possible $\boxed{p(X|C) = \prod_i p(X_i|C)}$

→ Must find other routes

- ↳ full cov matrix (Σ)
- ↳ Hidden markov model
- ↳ Gaussian Bayes Net

} can be arbitrarily complex

LDA

↳ Assumptions: - data for each feature is gauss distributed. ||

- (only consider bin class ie 1/0)

If we are checking

$$\frac{p(X|C=1)\,p(C=1)}{p(X)} > \frac{p(X|C=0)\,p(C=0)}{p(X)}$$

then

$$p(X|C=1)\,p(C=1) > p(X|C=0)\,p(C=0)$$

↳ plug in gauss likelihood for $p(x|c=i)$

$i \in \{0, 1\}$

↳ then note that $p(c=1) = 1 - p(c=0)$

↳ & take the log likelihoods

↳ move to left hand side

↳ simplify & combine

↳ result will have 3 parts & depends on x

$$\times$$

$$\left.\begin{array}{l} \text{①}\ x^T \frac{1}{2}(\Sigma_0^{-1} - \Sigma_1^{-1})x + (\mu_1^T \Sigma_1^{-1} - \mu_0^T \Sigma_0^{-1})x - \\ \text{②}\ \frac{1}{2}\mu_1^T \Sigma_1^{-1}\mu_1 + \frac{1}{2}\mu_0^T \Sigma_0^{-1}\mu_0 + \\ \text{③}\ \log\alpha - \log(1-\alpha) + \mu_1 - \mu_0 \end{array}\right\} \gtrless 0$$

① Quadratic term
② Linear term
③ const. term

} Quad discriminant analysis

$$x^T A x + w^T x + b > 0$$

↑ way simpler form

where

$$A = \frac{1}{2}(\Sigma_0^{-1} - \Sigma_1^{-1})$$

$$w^T x = (\mu_1^T \Sigma_1^{-1} \mu_1 - \mu_0 \Sigma_0^{-1} \mu_0)$$

$$b = -\frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_0^T \Sigma_0^{-1} \mu_0 + k_1 + k_0 + \log\alpha - \log(1-\alpha)$$

if Σ is diagonal then this simplifies w/ quad term vanishing

↳ if assumptions about data is true then bayes clf is optimal.

Generative vs Discriminative Models

↳ probability, predict $P(C|X)$

↳ Logit does this directly
also KNN

↳ Generative

↳ Calculates $P(X|C)$ & reverses prob w/ bayes rule ↑ for each C

Generative

↳ each var modeled explicitly w/ high level of explainability

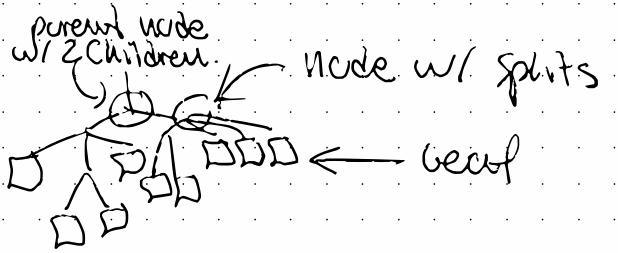
Discriminative

↳ Classifies better

→ Decision Trees

↳ makes predictions based on decisions

very intuitive



Challenges: how to make splits
how complex to make tree?

Splits are only possible orthogonal to axes

↳ previous could generate boundaries to arbitrary angle to axes

How to make splits?

↳ max entropy (split which maximizes information gain

$$H(p) = -\sum_x p(x)\log(p(x))$$ ← relates to variance

for binary clf

$$H(p) = -p\log(p) - (1-p)\log(1-p)$$

$$\frac{\partial H(p)}{\partial p} = 0 \leftarrow \text{gives optimal value}$$

"entropy is a measure of how much info we get from making a split"

Algorithms for Maximising entropy gain

ID3:

for each c in cols:

↳ find c that best splits data

↳ repeat for each split until all c is used

Others:

↳ use c multiple times

Pseudo code

```
def fit(X, y)
    Best ig = 0
    Best attr = None
    for c in Cols:
        cond = find split(X, y, c)
        y_left = y[X[c] # meets cond
        y_right = y[X[c] # ~meets cond
        i_gain = H(y) - p(left) H(y_left) - p(right) H(y_right)
        if best:
            save to find vars
    :
```

→ Perceptron

Setup
↳ Binary classification
w/ targets -1 / +1

Predict.

calc $w^T x + b$ → if = 0 df = ?
if > 0 df = 1
if < 0 df = -1

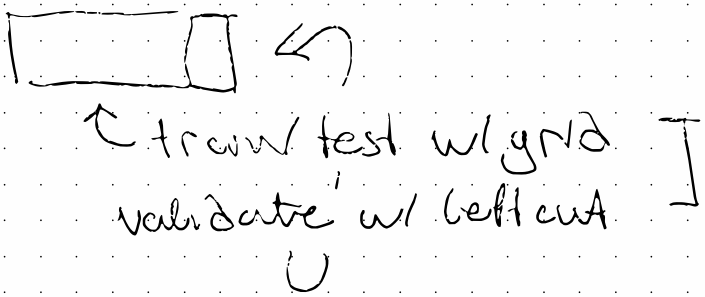
train/fit

```
w = random vector
b = 0
eta = 0.1 # or 1 or .001 etc
for i in n_epochs:
    get misclassifications (try, except break)

    x, y ← get 1 mis df

    w = w + eta·y·x
```

→ Practical Machine Learning

↳ for each algorithm, what is best for hyperparameter tuning?

↳ use CV w/ grid search (or variant of it)



↑ train/test w/ grid validate w/ left out

→ "Generalization"

→ Parameter space for gs?

↳ depends on compute power

↳ experience/trial & error

→ feature extraction & selection

↳ vital for domain problem

↳ 1 feature (random) makes it super easy to do classification

→ requires knowledge of data (domain knowledge)

↳ not all features are useful & can lead to overfitting

various algorithms to choose

↳ select all & elim

↳ select 1 & build

↳ try all combos

↳ keep only best

...

etc

Other methods:

↳ PCA

↳ Select features w/ no corr

↳ Problem: only a lin. transform

→ Classic algos vs DL

↳ Some algos train & predict faster than others

↳ Depends on usecase

↳ Decision boundaries

↳ Depends on variance/bias tradeoff (also interpretable)

→ Multiclass Classifications

→ 1 vs rest → many 1 vs 1

↳ cumbersome & more difficult

→ ROC/AUC

↳ if output is a probability then make classification based on threshold

Auc = performance

point closest to (0, 1) is best threshold to use.