7 EVALUATION METRICS

$$\text{ACCURACY} = \frac{\text{TRUE +VE + TRUE -VE}}{\text{ALL}}$$

MEDICAL MODEL

$$\frac{P_{TRUE}}{P_{TRUE} + P_{FALSE}}$$

- FALSE +VE OK
- FALSE -VE NOT OK
- << FIND ALL THE SICK PEOPLE >>

HIGH RECALL

SPAM DETECTOR

$$\frac{P_{TRUE}}{P_{TRUE} + P_{FALSE}}$$

- FALSE +VE NOT OK
- FALSE -VE OK
- << MARKED SPAM? BETTER BE SPAM! >>

HIGH PRECISION

F1 SCORE - COMBINES PRECISION (P) AND RECALL (R)

$$F_1 = \frac{2PR}{P+R}$$

HARMONIC MEAN

F_β - WEIGHT PRECISION OR RECALL MORE.

$\beta < 1$ WEIGHS PRECISION MORE

$\beta > 1$ WEIGHS RECALL MORE

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R}$$

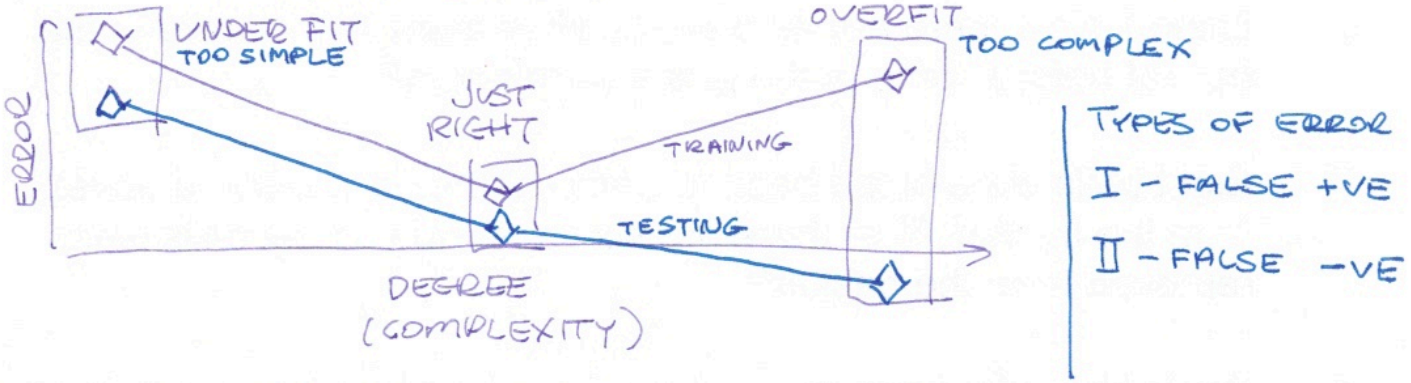
8 MODEL SELECTION

- TYPES OF ERRORS

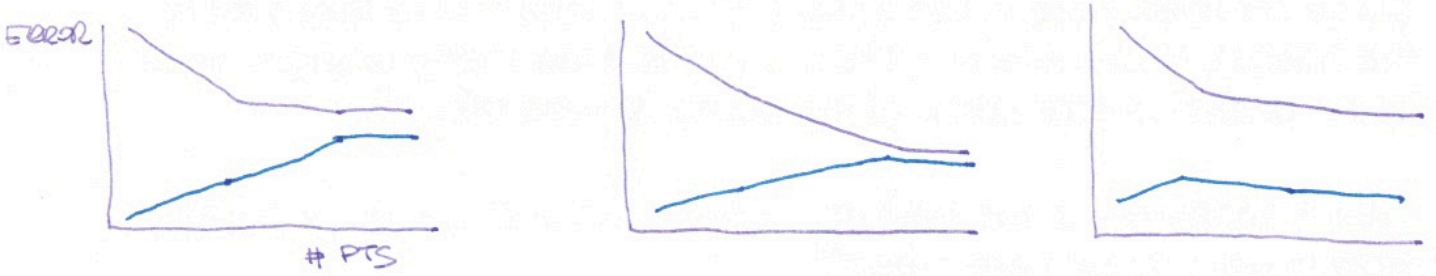
GODZILLA w/ FLYSWATTER
OVERSIMPLIFYING
UNDERFITTING
DOES NOT DO WELL IN TRAINING
ERROR DUE TO BIAS

FLY w/ BAZOOKA
OVER COMPLICATING
OVERFITTING
GOOD IN TRAINING, BAD TESTING
ERROR DUE TO VARIANCE

- MODEL COMPLEXITY GRAPH



- CROSS VALIDATION - SET APART DATA FOR FINDING MODEL PARAMS
- K-FOLD CROSS VALIDATION - TRAIN K TIMES w/ DIFFERENT 1/K PORTION OF DATA FOR TESTING. HELPS PREVENT OVERFITTING
  sklearn.model_selection.KFold
- LEARNING CURVES



HIGH BIAS
UNDERFITTING

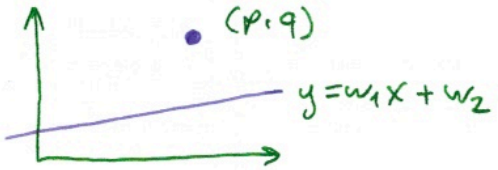
GOOD MODEL

HIGH VARIANCE
OVERFITTING

- GRID SEARCH - USED TO FIND BEST HYPERPARAMETERS
  SKLEARN: sklearn.model_selection.GridSearchCV

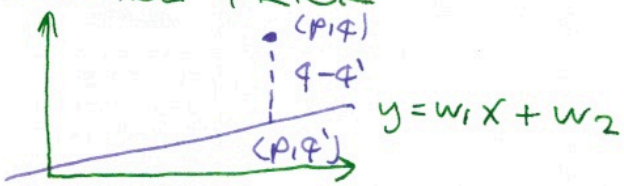
11 LINEAR REGRESSION $y = w_1 x + w_2$

- SUPERVISED MACHINE LEARNING
  - CLASSIFICATION — YES/NO
  - REGRESSION — HOW MUCH?
- LEARNING RATE: $\alpha < 1$ AVOIDS TAKING TOO-LARGE STEPS
- ABSOLUTE TRICK



$$y = (w_1 + \alpha p)x + (w_2 + \alpha)$$

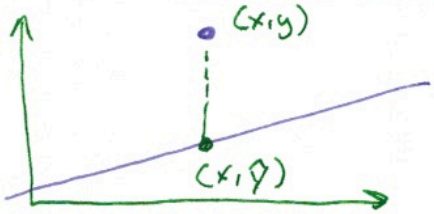
- SQUARE TRICK



$$y = (w_1 + \alpha p(q - q')) + (w_2 + \alpha(q - q'))$$

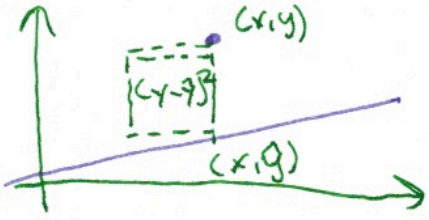
- GRADIENT DESCENT $w_i \longrightarrow w_i - \alpha \frac{\partial}{\partial w_i} \text{Error}$
- ERROR FUNCTIONS

MEAN ABSOLUTE ERROR



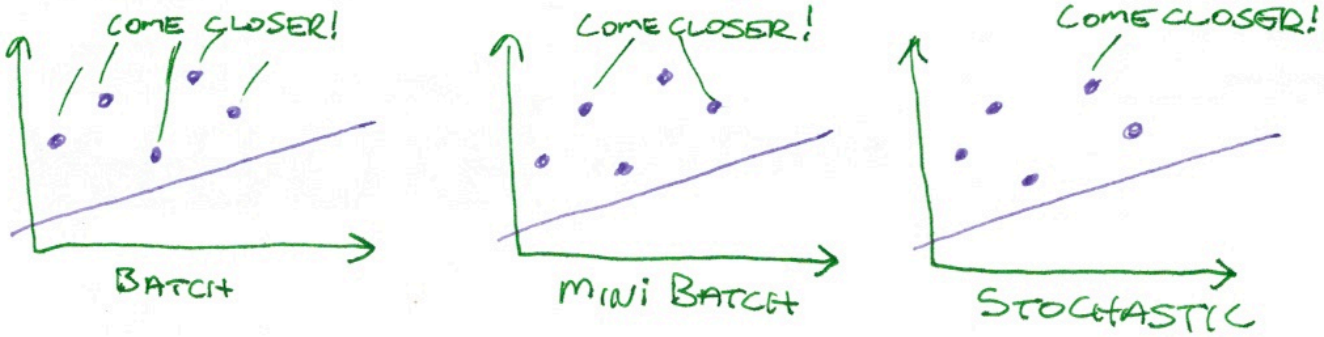
$$\text{Error} = \frac{1}{n} \sum_{i=1}^{n} |y - \hat{y}|$$

MEAN SQUARED ERROR



$$\text{Error} = \frac{1}{2m} \sum_{i=1}^{m} (y - \hat{y})^2$$

- TYPES OF GRADIENT DESCENT



11 LINEAR REGRESSION (CONTINUED)

- LINEAR REGRESSION IN SCIKIT LEARN

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_values, y_values)
```

- HIGHER DIMENSIONS

  n DIMENSIONAL SPACE → PREDICTION IS n-1 DIMENSIONAL HYPERPLANE

  $\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_{n-1} x_{n-1} + w_n$

- LINEAR REGRESSION HAS A CLOSED-FORM SOLUTION FOR WEIGHTS W, VARIABLES X AND LABELS y

  $W = (X^T X)^{-1} X^T y$

  EXPENSIVE IN PRACTICE - STILL USE GRADIENT DESCENT

- LINEAR REGRESSION WARNINGS
  1. WORKS BEST WHEN DATA IS LINEAR.
  2. SENSITIVE TO OUTLIERS.

- POLYNOMIAL REGRESSION
  - NON-LINEAR DATA
  - SAME GRADIENT DESCENT METHOD
- REGULARIZATION
  - ADD TERMS TO ERROR TO PENALIZE COMPLEXITY
  - L_1 - ADD ABSOLUTE VALUE OF FIT COEFFICIENTS
  - L_2 - ADD SQUARES OF FIT COEFFICIENTS
  - ADJUST λ TO ADJUST PUNISHMENT FOR COMPLEXITY
    LARGE λ - PUNISH COMPLEXITY
  - WHICH

L_1	L_2
COMPUTATIONALLY INEFFICIENT	EFFICIE
SPARSE OUTPUTS	NON-SPARSE OUTPUTS
FEATURE SELECTION	

12 PERCEPTRONS (CLASSIFICATION)

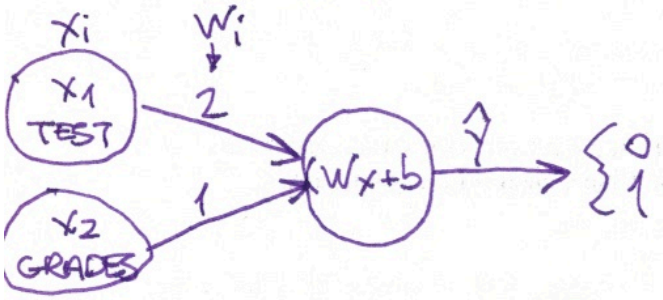
- CLASSIFICATION

LINE BOUNDARY $w_1x_1 + w_2x_2 = 0$ $\quad Wx + b = 0$

PREDICTION $\hat{y} = \begin{cases} 1 & \text{if } Wx+b \geq 0 \\ 0 & \text{if } Wx+b < 0 \end{cases}$

N-DIMENSIONS: BOUNDARY IS $n-1$ DIMENSIONAL HYPERPLANE.

- PERCEPTRON



- PERCEPTRON TRICK
  - GOAL: SPLIT DATA CLASSES W/ ONE LINE
  - POINTS ON THE WRONG SIDE WANT THE LINE TO "COME CLOSER".
  - TRICK
    - EX: $3x_1 + 4x_2 - 10$ BAD POINT AT $(4,5)$

      $3 \quad 4 \quad -10$ LEARNING RATE

      $(4 \quad 5 \quad 1) \times 0.1$

      - − IF MISCLASSIFIED IN +VE ARE
      - \+ IF MISCLASSIFIED IN −VE AREA

- PERCEPTRON ALGORITHM
  1. START W/ RANDOM WTS $w_1, \ldots, w_n, b$
  2. FOR EVERY MISCLASSIFIED PT $(x_1, \ldots, x_n)$
     - 2.1 ~~$x_i =$~~ $w_i = w_i + \alpha x_i \quad \forall x \in 1, \ldots, n$ } PREDICTION = 0, +VE PT IN −VE AREA
     - ~~2.2~~ $b = b + \alpha$
     - 2.2 PREDICTION = 1 −VE PT IN +VE AREA

       $w_i = w_i - \alpha x_i$

       $b = b - \alpha$

[13] Decision Trees

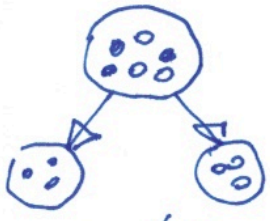
- Intro: classify by asking questions of the data. Need to know what question (data) tells us the most
- Entropy
  - m red balls, n blue balls

$$\text{Entropy} = -\frac{m}{m+n}\log_2\left(\frac{m}{m+n}\right) - \frac{n}{m+n}\log_2\frac{n}{m+n}$$

Let $p_1 = \frac{m}{m+n}$, $p_2 = \frac{n}{m+n}$ $\Rightarrow S = -p_1\log_2 p_1 - p_2\log_2 p_2$

$$S = -\sum_{i=1}^{n} p_i \log_2 p_i$$

  - Information Gain



$$\text{Information Gain} = S_{\text{parent}} - \left(\frac{m}{m+n}S_{\text{child}_1} + \frac{n}{m+n}S_{\text{child}_2}\right)$$

Build decision trees by maximizing information gain.

- Random Forest
  - Decision trees prone to overfitting
  - Build multiple trees from randomly selected features and have them vote
- Hyperparameters
  - Max depth
  - Min samples per leaf
  - Min samples per split
  - Max number of features
- Decision trees in sklearn: DecisionTreeClassifier

14 NAIVE BAYES

PROBABILISTIC ALGORITHM BASED ON CONDITIONAL PROBABILITY

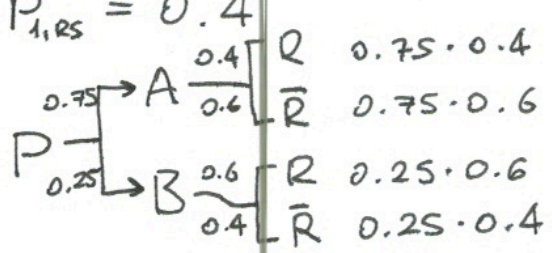
EASY TO IMPLEMENT, FAST

- BAYES THEOREM
  - GUESS THE PERSON
    - w/o INFO $P=0.5$ FOR EACH (PRIOR)
    - WE SEE A RED SWEATER $P_{1,RS}=0.4$ $P_{2,RS}=0.6$ (POSTERIOR)
  - KNOWN → INFERRED

$P(A)$
$P(R|A)$ $\Rightarrow P(A|R)$

  - ADD MORE INFO SCHEDULE: $P_{1,IN}=0.75$ $P_{2,IN}=0.25$

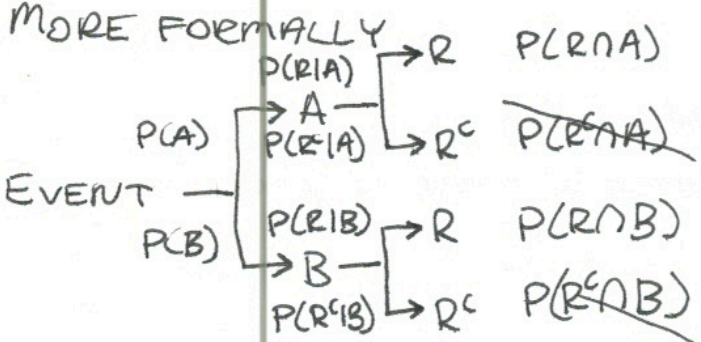
$P_{1,RS}=0.4$



PERSON IN RED SO $\bar{R}$ NOT POSSIBLE

$$P(A|R)=\frac{0.75\cdot 0.4}{0.75\cdot 0.4+0.25\cdot 0.6}=0.67$$

  - MORE FORMALLY



$$\Rightarrow P(A|R)=\frac{P(A)P(R|A)}{P(A)P(R|A)+P(B)P(R|B)}$$

  - BAYESIAN LEARNING – SPAM EXAMPLE
  - NAIVE BAYES ALGORITHM
    - NAIVE – ASSUME EVENTS ARE INDEPENDENT

$P(\text{spam}|\text{'easy'},\text{'money'}) \propto P(\text{'easy'}|\text{spam})P(\text{'money'}|\text{spam})P(\text{spam})$

$P(\text{ham}|\text{'easy'},\text{'money'}) \propto P(\text{'easy'}|\text{ham})P(\text{'money'}|\text{ham})P(\text{ham})$

COMPUTE THE TWO ABOVE AND NORMALIZE TO ONE.

USING GRIDSEARCHCV

1. DEFINE DICTIONARY W/ PARAMETERS TO TEST:
   PARAMS = {"PARAM_NAME": [VAL1, VAL2, ...], ...}
   * BE CAREFUL WHEN USING FRACTIONS IN PARAMETER GRID
2. CREATE MODEL/CLASSIFIER
   model = DecisionTreeClassifier()
   USE FIXED random_state FOR REPEATABILITY
3. CREATE A SCORER. USE make_scorer TO CONVERT A METRIC TO A SCORER,
   scorer = make_scorer(accuracy_score)
4. SET UP THE GRID SEARCH.
   clf = GridSearchCV(model, param_grid, cv=4, n_jobs=8, scoring=scorer)
5. RUN THE GRID SEARCH.
   clf.fit(X_train, y_train)
6. PREDICT AND CALCULATE ACCURACY.
   y_train_pred = clf.predict(X_train) } USES BEST FIT.
   y_test_pred = clf.predict(X_test) }
   train_accuracy = accuracy_score(y_train, y_train_pred)
   test_accuracy = accuracy_score(y_test, y_test_pred)

BAYESIAN INFERENCE TUTORIAL

- DATA SET: SMS SPAM COLLECTION
  - SPAM/HAM, MESSAGE
  - USE pd.Series.map to map "spam" ↦ 1, "ham" ↦ 0
- BAG OF WORDS
  - sklearn.feature_extraction.text.CountVectorizer
  - MANUALLY
    1. str.lower()
    2. str.translate(str.maketrans("", "", string.punctuation))
    3. str.split()
    4. collections.Counter → DICT OF WORD OCCURRENCES

       Counter(s) ↦ DICT OF WORD FREQUENCIES
  - sklearn CountVectorizer
    1. cv = CountVectorizer() LOWER CASE, STRIP PUNCT, NO STOPS
    2. cv.fit(docs)
    3. cv.transform(docs).toarray() ↦ CONVERT FREQS TO DOC MATRIX
- BAYES THEOREM IMPLEMENTATION

  P(D) = ~~PR~~ 0.01 PROB OF DIABETES

  P(+) = PROB +VE TEST RESULT

  P(−) = " −VE " "

  P(+|D) = 0.9 SENSITIVITY = TRUE +VE RATE

  P(−|~D) = 0.9 SPECIFICITY = TRUE −VE RATE
- EVALUATION
  - ACCURACY = CORRECT PREDICTIONS / TOTAL PREDICTIONS
  - PRECISION = WHAT PORTION OF MSGS MARKED SPAM ARE SPAM

    P = TRUE +VE / (TRUE +VE + FALSE +VE)
  - RECALL (SENSITIVITY) = WHAT PORTION OF SPAM WAS MARKED AS SPAM

    R = TRUE +VE / (TRUE +VE + FALSE NEGATIVE)

15 SUPPORT VECTOR MACHINES (SVMs)

- TRIES TO MAXIMIZE DISTANCE BETWEEN BOUNDARY LINE AND DATA POINTS
- ERROR = CLASSIFICATION ERROR + MARGIN ERROR
  - MISCLASSIFICATIONS PUNISHED ∝ DISTANCE FROM DIVIDING LINE
  - CLASSIFICATION ERROR: $Wx+b = c$ → ERROR TERM
    SUM DISTANCES OF POINT FROM MARGIN
  - MARGIN ERROR: ERR ∝ 1/SIZE MARGIN $= \frac{2}{|W|}$ = MARGIN
    SO ERROR = $|W|^2$
- ERROR FUNCTION = MARGIN ERROR + CLASSIFICATION ERROR
- C PARAMETER – COEFFICIENT FOR CLASSIFICATION ERROR
  - SMALL C → LARGE MARGIN, POSSIBLE CLASSIFICATION ERRORS
  - LARGE C → SMALL MARGIN, CLASSIFIES PTS WELL
  - HYPERPARAMETER
- POLYNOMIAL KERNEL
  - A LINE MAY NOT BE ABLE TO DIVIDE CLASSES
  - KERNEL TRICK – NONLINEAR FNS OR ADDITIONAL DIMENSIONS

KERNEL TRICK

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma_2}}$$

2-D
(x, y)
$(2, 3)$

→

5-D
(x, y, x^2, xy, y^2)
$(2, 3, 4, 6, 9)$

DEGREE-2 POLYNOMIAL ← 4D BOUNDARY HYPERPLANE

DEGREE OF KERNEL IS A HYPERPARAMETER

- RADIAL BASIS FUNCTION (RBF) KERNELS
  - USE LINEAR COMBINATIONS OF RBFs TO SEPARATE CLASSES
  - HYPERPARAMETER γ DETERMINES WIDTH OF RBF
    $\gamma = \frac{1}{2\sigma^2}$
- SVMs in sklearn
  - sklearn.svm.SVC()
  - HYPERPARAMETERS: C, KERNEL, DEGREE, GAMMA

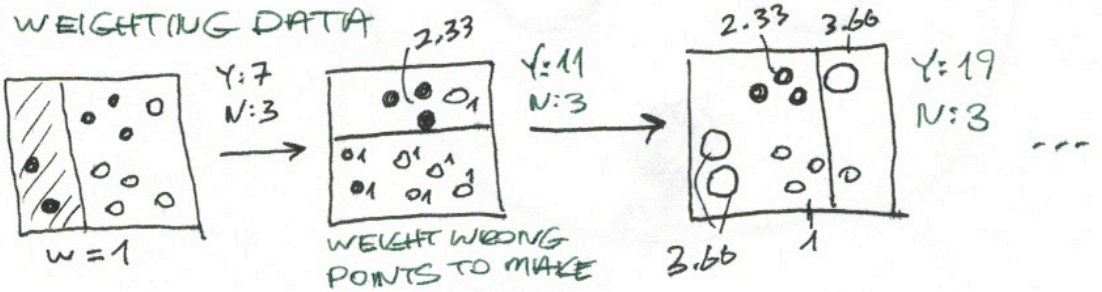
linear, poly, rbf — MAX DEGREE FOR poly — RBF WIDTH

16 ENSEMBLE METHODS - USING DIFFERENT MODELS TOGETHER

- METHODS
  - BAGGING (BOOTSTRAP AGGREGATING)
  - BOOSTING

COMBINE WEAK LEARNERS ⇓ STRONG LEARNER

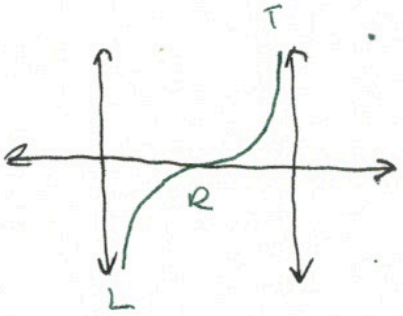
- BAGGING
  - WEAK LEARNER - ONLY SLIGHTLY BETTER THAN RANDOM
  - TRAIN MANY WEAK LEARNERS ON DIFFERENT SUBSETS OF DATA
  - COMBINE WEAK LEARNERS BY VOTING/AVERAGING
- ADABOOST
  1. TRAIN WEAK LEARNER TO SOME MINIMUM ACCURACY
  2. TRAIN 2ND LEARNER - PENALIZING POINTS MISCLASSIFIED BY FIRST LEARNER MORE
  3. CONTINUE W/ MORE LEARNERS
  4. COMBINE MODELS
- WEIGHTING DATA



- WEIGHTING THE MODELS

{ TRUTHFUL +VE / RANDOM (50/50) 0 / LIAR −VE }

$$\text{WEIGHT} = \ln\left(\frac{\text{accuracy}}{1-\text{accuracy}}\right) = \ln\left(\frac{\text{correct}}{\text{wrong}}\right)$$



FOR ACCURACY = 1 OR 0 – JUST USE THOSE MODELS

- COMBINING THE MODELS

CALCULATE WEIGHTS USING WEIGHTED DATA RESULTS

- ADABOOST IN sklearn
  - sklearn.ensemble.AdaBoostClassifier()
  - HYPERPARAMETERS
    - base_estimator: WEAK LEARNER MODEL
    - n_estimators: MAX NUMBER OF WEAK LEARNERS

QUESTION 4 SUPERVISED LEARNING ASSESSMENT

PROJECT: FINDING DONORS FOR CHARITY ML

- OVERVIEW
  - EXPLORE DATA
  - TRANSFORM DATA TO WORKABLE FORMAT
  - EVALUATE SEVERAL LEARNERS
  - OPTIMIZE ONE OF THE LEARNERS
  - TEST
- GOAL: MAXIMIZE DONATIONS, MINIMIZE NUMBER OF LETTERS SENT.
- SUBMISSION
  - COMPARE ALL SECTIONS AGAINST RUBRIC
  - SUBMIT IPYNB NOTEBOOK AND EXPORTED report.html
  - SUBMIT IN finding_donors FOLDER IN GITHUB REPO.
- MEASURING PERFORMANCE

$$\text{ACCURACY} = \frac{TP + TN}{N_{PRED}}$$

$$\text{PRECISION } P = \frac{TP}{TP + FP} \qquad \text{RECALL } R = \frac{TP}{TP + FN}$$

$$F_\beta = (1+\beta^2)\frac{P \cdot R}{\beta^2 P + R}$$

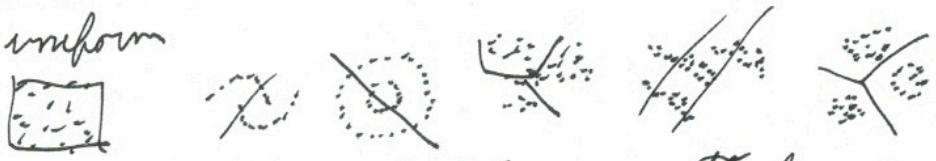
19 CLUSTERING

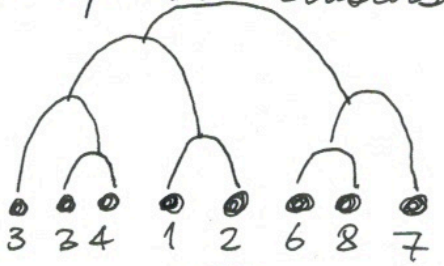
- Unsupervised learning – no labels but can still group things
  - clustering
  - dimensionality reduction
- K-means: how many clusters?
  - Steps: 1. assign 2. Optimize
    1. assign points to the nearest class center
    2. Optimize cluster center to minimize sum of square of distances to points in the cluster
    - Initial positions of centroids is important!
- sklearn.cluster.KMeans
  - parameters
    - n_clusters (8) usually need to set this
    - max_iter (300)
    - n_init (10) number of times to run w/ different centroid seeds

21 Hierarchical and Density-Based Clustering

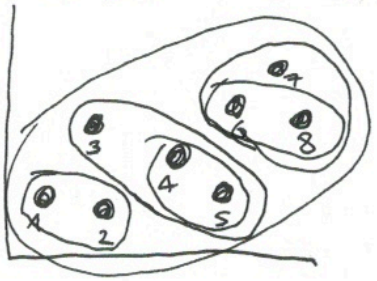
- K-means considerations
  - Need to know number of clusters
  - clusters are hyperspherical - can't do non-spherical clusters
  - data sets

uniform

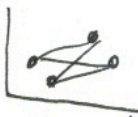




- Overview of other clustering methods
  - Hierarchical clustering
  - Density clustering (DBScan)



- Hierarchical clustering: single-link
  - distance: smallest distance between any two points in two clusters
  - handles two arcs, two rings, distinct clusters as well or better than K-means
- Hierarchical clustering: three other methods (agglomerative clustering)
  - complete link: farthest distance between any two points in two clusters
  - average link: average distance between all points in two groups
  - Ward's method $\Delta(A,B) = C_1^2 + C_2^2 + C_3^2 + C_4^2 - A_1^2 - A_2^2 - B_{1,2}^2 + B_1^2$



- Hierarchical clustering: implementation
  - sklearn.cluster.AgglomerativeClustering

```
clust = AgglomerativeClustering(n_clusters=3, linkage="ward")
labels = clust.fit_predict(X)
```

  - Dendograms - use scipy.cluster.hierarchy
  - Advantages
    - Resulting hierarchical representation is very informative
    - Visualization
    - Good at modeling real hierarchical relations
  - Disadvantage
    - Sensitive to noise, outliers • expensive

- Density-based clustering: DBSCAN
  1. Select a pt
  2. Look for pts w/in ϵ
     - no pts → noise
     - less than MinPts → noise
     - ≥ MinPts → cluster w/ core, border pts
- Implementation

```
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.5, min_samples=5)
db.fit(X)
db.labels_  # clusters, -1 = noise
```

- DBSCAN examples and applications
  - Advantages
    - Don't need to specify number of clusters
    - Flexibility in shapes, sizes of clusters
    - Handles noise and outliers
  - Disadvantages
    - Border pts reachable from 2 clusters
    - Faces difficulty w/ clusters of varying densities

22 Gaussian Mixture Models

- soft clustering - all points in all clusters - membership
- cluster analysis
- cluster validation

- gmm clustering

	w/in
• based on Gaussian (normal) dist	1σ 68%
parameters (1D): $\mu \equiv$ mean $\sigma \equiv$ std dev.	2σ 95%
• fit pt density w/ Gaussians	3σ 99%

- Expectation-maximization for Gaussian mixtures
  1. Init K Gaussian dists (can use k-means)
  2. soft-cluster data - "expectation" *
  3. Re-estimate Gaussians - "maximization" **
  4. Eval log-likelihood to check for convergence ***

* $$E[z_{iA}] = \frac{N(x_i|\mu_A,\sigma_A^2)}{N(x_i|\mu_A,\sigma_A^2)+N(x_i|\mu_B,\sigma_B^2)}$$

** $$\mu_A = \frac{\sum_{i=1}^{N} E[z_{ij}]x_i}{\sum_{i=1}^{N} E[z_{ij}]}$$

$$\sigma_A^2 = \frac{\sum_{i=1}^{N} E[z_{iA}](x_i-\mu_A^{new})(x_i-\mu_A^{new})^T}{\sum_{i=1}^{N} E[z_{iA}]}$$

*** $$\ln p(X|\mu,\sigma^2) = \sum_{i=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k N(x_i|\mu_k,\sigma_k^2)\right)$$

- Implementation

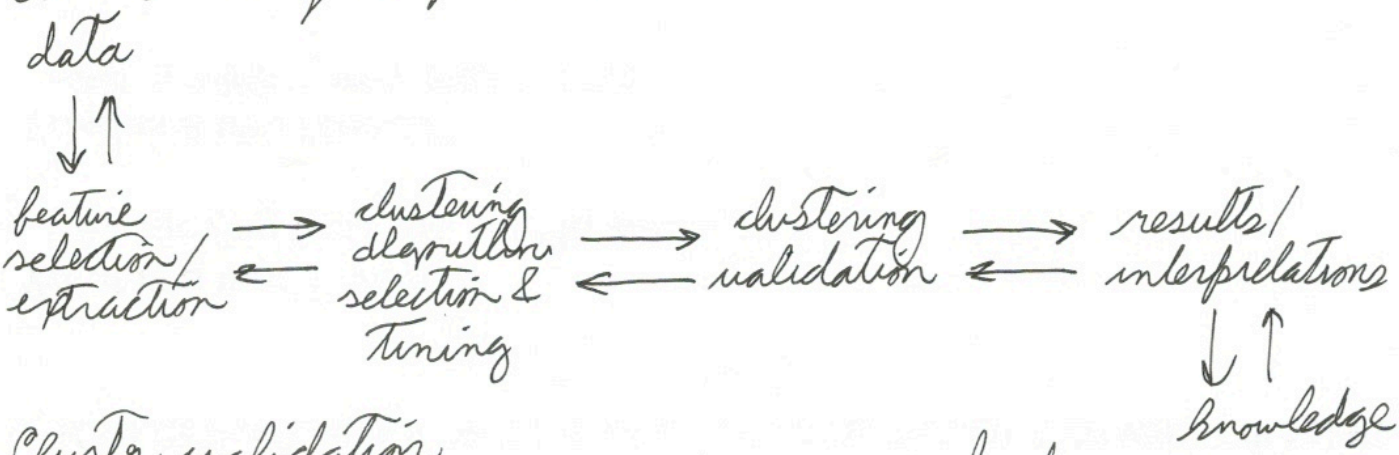
```
gmm = sklearn.mixture.GaussianMixture(n_components=3)
gmm.fit(X)
clustering = gmm.predict(X)
    ↳ array of cluster memberships
```

- Examples/advantages

• soft clustering	pros
• cluster shape flexibility	pros
• sensitive to init values	cons
• local optima	cons
• slow convergence	cons

22 Gaussian Mixture Models

- Cluster analysis process



- Cluster validation
  1. External indices (labels) — compactness
  2. internal indices (no labels)
  3. relative indices — separability
  - External indices: adjusted rand index
  - Internal indices: silhouette coefficient $[-1, 1]$

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

$a_i \equiv$ avg dist in cluster
$b_i \equiv$ avg distance closest cluster

Maximize S_i to find K.
Never use this to measure DBSCAN

23 Feature Scaling

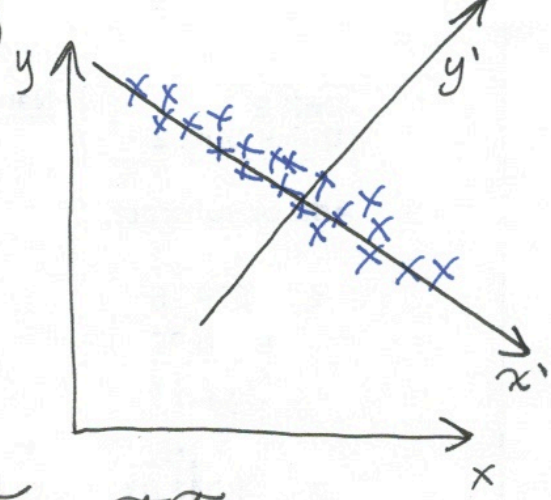
- Sometimes must rescale so all features have equal weight
- min-max scaler

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad 0 \leq x \leq 1$$

  - pro: fixed interval
  - con: outliers ruin scaling

- Implementation
  - sklearn.preprocessing.MinMaxScaler

24 Principal Component Analysis (PCA)



- axes shifts and rotations
- measurable vs latent

square footage	size
no. of rooms	
school ranking	neighborhood
neighborhood safety	

- principal component - composite feature that more directly probes the underlying phenomenon
  used for: dimensionality reduction, unsupervised learning
- How to determine the principal component?
  principal component of a data set is the direction that has the maximum variance
- Maximal variance and information loss
- Review / definition of PCA
  - transform input features to principal components
  - use principal components as new features
  - PCs are directions in data that maximize variance (minimize info loss)
  - more variance along PC → higher ranked PC
  - max. no of PCs: no. of input features
- Implementation

```
pca = sklearn.decomposition.PCA(n_components=2)
pca.fit(data)
pca.explained_variance_ratio_
pca.components_[
```

- When to use PCA
  - latent features
  - dimensionality reduction
  - reduce noise
  - preprocessing for other algorithms

16 Random Projection and ICA - Dimensionality Reduction

- Random Projection
  - good for high-dimensional problems
  - computationally efficient
  - $X^{RP}_{k \times N} = R_{k \times d} X_{d \times N}$ d dims to k dims
  - Johnson-Lindenstrauss lemma - can project to lower dim space largely preserving distance between pts
  - Implementation

```
from sklearn.random_projection import SparseRandomProjection
rp = SparseRandomProjection()   epsilon or n_components
new_X = rp.fit_transform(X)
```

- Independent Component analysis
  - Extract independent sources from a mixed sample
    << blind source separation >>
  - FastICA
    1. center, whiten X
    2. choose initial random weight matrix $W_1, W_2, \ldots, W_n$
    3. estimate W_i containing vectors
    4. decorrelate W
    5. repeat from 3 until convergence
  - ▽ Need as many observations as signals we are trying to separate.
  - Implementation

```
from sklearn.decomposition import FastICA
X = list(zip(signal_1, signal_2, signal_3))
ica = FastICA(n_components=3)
components = ica.fit_transform(X)
                ↳ independent components.
```