ECE M146 Introduction to Machine Learning

Lecture 17 - Spring 2021

Prof. Lara Dolecek

ECE Department, UCLA

Today's Lecture

Recap:

- Unsupervised Learning: K-means and PCA

New topic:

- Mixture modeling
- Expectation maximization; soft K-means

Recap: K-means

- We have already seen two popular instances of methods for unsupervised learning.
- K-means is an iterative method for clustering
  - Key idea: identify cluster representatives (prototype) and assign each data point to the cluster of the closest prototype, so that the sum of all distances from data points to their prototypes is minimized.
  - How: iterate between finding the closest prototype based on the given prototypes, for each data point, and adjusting the prototype position based on the points assigned to its cluster.

Recap: PCA

- We have already seen two popular instances of methods for unsupervised learning.
- PCA is a projection method for dimensionality reduction
  - Key idea: project onto the dimension of the highest sample variance as this is the most informative dimension.
  - How: formulate a constrained optimization problem and then maximize a Lagrangian. Use linear algebra to conclude that this dimension actually corresponds to the eigenvector of the largest eigenvalue.

Today's Lecture

Recap:

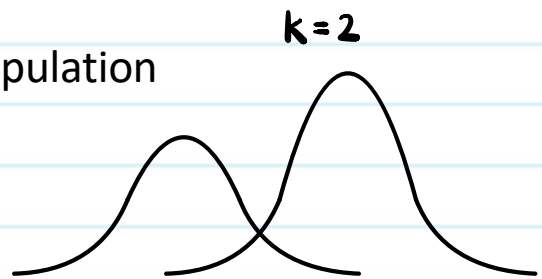
- Unsupervised Learning: K-means and PCA

New topic:

- Mixture modeling
- Expectation maximization; soft K-means

Mixture modeling

- Mixture modeling is useful for modeling distributions that have multiple peaks.
- Example: height distribution in a population



- Mathematically:

$$P(X) = \sum_{k=1}^{K} \pi_k \cdot P(X \mid \text{component } k \text{ is chosen})$$

$$\sum_{k=1}^{K} \pi_k = 1$$

Gaussian Mixture Modeling (GMM)

- Can be viewed as a universal approximator for any distribution provided the number of components is sufficiently large.
- Mathematically:

$$P(X) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(X; \mu_k, \Sigma_k)$$

- Interpretation: pick a component *k* with probability π_k. Then generate a sample according to the distribution $\mathcal{N}(\mu_k, \Sigma_k)$.

Recall: Multivariate Gaussian distribution

- We have already studied this distribution.
- Where ?
- When we studied Gaussian Discriminant Analysis, both LDA and QDA.
  - This was an example of generative modeling; classification; supervised learning.

here: new application of unsupervised learning

Recall: Multivariate Gaussian Distribution

- Interpretation of the parameters: mean vector and covariance matrix.

X is dim D

$X \sim \mathcal{N}(\mu, \Sigma)$

$$P(X) = \frac{1}{(2\pi)^{D/2}(\det\Sigma)^{1/2}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

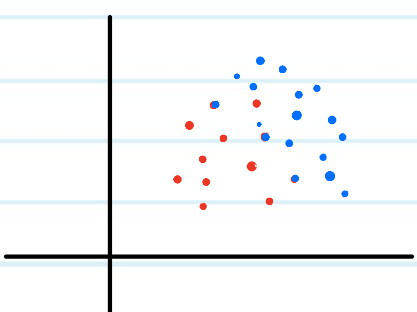
$$\mu = \begin{bmatrix} E(X_1) \\ E(X_2) \\ \vdots \\ E(X_D) \end{bmatrix} \qquad \Sigma_{ij} = \text{cov}(X_i, X_j)$$

(Gaussian) Mixture Modeling

- This mixture modeling also allows for “soft” a.k.a. partial cluster membership.

- Instead of using “hard” indicators with cluster membership being strictly 0 or 1, i.e., each data point belongs to one and only one cluster, allow for fractional membership.

- Picture:



Today's Lecture

Recap:

- Unsupervised Learning: K-means and PCA

New topic:

- Mixture modeling
- Expectation maximization; soft K-means

Set-up

- Say we have N data points, which we seek to organize into K clusters.
- Data likelihood:

$$P(X) = \prod_{n=1}^{N} P(X_n) = \prod_{n=1}^{N} \left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(X_n; \mu_k, \Sigma_k) \right)$$

- Goal is to figure out what the parameters are:

$$\pi_k, \mu_k, \Sigma_k$$

$$1 \leq k \leq K$$

Set-up

- How ? $P(X) = \prod_{n=1}^{N} P(X_n) = \prod_{n=1}^{N} \left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(X_n; \mu_k, \Sigma_k) \right)$

take log:

$$\sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(X_n; \mu_k, \Sigma_k) \right)$$

- When we encountered problems in the past that dealt with likelihood maximization w.r.t. parameters, the strategy was to write log likelihood, and set the derivatives to zero. But can't to it here.

Expectation-maximization (EM)

- EM is a general, **iterative** method for finding a maximum (log) likelihood solution, in the presence of hidden (latent) variables.

- Usage beyond clustering.

```
EM has uses in engineering, finance,
medicine, etc.
```

Define responsibilities

let z_n be the cluster membership of data point x_n

$P(z_n = k) = \gamma_{nk}$ is the probability that x_n is in cluster k

$P(z_n = k)$ is called "responsibility" (because it captures how much is cluster k "responsible" for x_n

$$\gamma_{nk} = \frac{\pi_k \cdot \mathcal{N}(x_n; \mu_k, \Sigma_k)}{\sum_{l=1}^{K} \pi_l \cdot \mathcal{N}(x_n; \mu_l, \Sigma_l))}$$

Expectation-maximization (EM)

- EM is a general, **iterative** method for finding a maximum (log) likelihood solution, in the presence of hidden (latent) variables.

- EM has two steps:

1. Estimates of the posteriors, called responsibilities:
   - Called responsibilities because they quantify how much is each cluster responsible in explaining a given data point.

2. MLE estimates of the parameter set:
   - Derivatives, as before.

EM – discussion

- If we knew true posteriors, we could plug them into the MLE estimates; if we knew MLE estimates, we could plug them into the posteriors.
- But we know neither!
- Solution: iterate.

- What does this approach remind you of ?

  `K-means for clustering`

Now, for the mathematical details

- Write log likelihood:

$$L = \sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n; \mu_k, \Sigma_k)\right)$$

- Responsibilities:

$$\gamma_{nk} = \frac{\pi_k \cdot \mathcal{N}(x_n; \mu_k, \Sigma_k)}{\sum_{l=1}^{K} \pi_l \cdot \mathcal{N}(x_n; \mu_l, \Sigma_l))}$$

Maximizing parameters

- If responsibilities were known, we can set the derivatives of the component parameters to zero.

- For the mean we get:

$$\frac{\partial L}{\partial \mu_l} = 0$$

$$\frac{\partial}{\partial \mu_l}\left[\sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n;\, \mu_k, \Sigma_k)\right)\right]$$

$$= \sum_{n=1}^{N} \frac{\pi_l \left(\frac{\partial}{\partial \mu_l} \mathcal{N}(x_n;\, \mu_l, \Sigma_l)\right)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n;\, \mu_k, \Sigma_k)}$$

Maximizing parameters, ctd.

- For the mean, we get:

$$\frac{\partial}{\partial \mu_\ell} \mathcal{N}(x_n;\, \mu_\ell, \Sigma_\ell) = \frac{\partial}{\partial \mu_\ell}\left[\frac{1}{(2\pi)^{D/2}(\det \Sigma_\ell)^{1/2}} e^{-\frac{1}{2}(x_n-\mu_\ell)^T \Sigma_\ell^{-1}(x_n-\mu_\ell)}\right]$$

$$= \mathcal{N}(x_n;\, \mu_\ell, \Sigma_\ell) \cdot \Sigma_\ell^{-1}(x_n - \mu_\ell)$$

$$\frac{\partial L}{\partial \mu_\ell} = \sum_{n=1}^{N} \frac{\pi_\ell \mathcal{N}(x_n;\, \mu_\ell, \Sigma_\ell) \cdot \Sigma_\ell^{-1}(x_n-\mu_\ell)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n;\, \mu_k, \Sigma_k)} = 0$$

$$\sum_{n=1}^{N} \gamma_{n\ell} \Sigma_\ell^{-1}(x_n - \mu_\ell) = 0$$

$$\sum_{n=1}^{N} \gamma_{n\ell}(x_n - \mu_\ell) = 0 \quad \Rightarrow \quad \hat{\mu}_\ell = \frac{\sum_{n=1}^{N} \gamma_{n\ell} x_n}{\sum_{n=1}^{N} \gamma_{n\ell}}$$

Maximizing parameters:

- For the covariance, we get:

$$\hat{\Sigma}_k = \frac{\sum_{n=1}^{N} \gamma_{nk}(x_n - \hat{\mu}_k)(x_n - \hat{\mu}_k)^T}{\sum_{n=1}^{N} \gamma_{nk}}$$

Maximizing parameters:

- For the mixing coefficients, we get:

$$\sum_{l=1}^{K} \pi_l = 1$$

$$L = \sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n; \mu_k, \Sigma_k)\right)$$

this is again a constrained optimization problem: set up Lagrangian

Maximizing parameters:

- For the mixing coefficients, we get:

Lagrange coefficient $\lambda \neq 0$

$$\sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n; \mu_k, \Sigma_k)\right) + \lambda\left(\sum_{k=1}^{K} \pi_k - 1\right)$$

set derivative w.r.t. π_l as 0: $\sum_{n=1}^{N} \frac{\mathcal{N}(x_n; \mu_l, \Sigma_l)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n; \mu_k, \Sigma_k)} + \lambda = 0$

$$\sum_{n=1}^{N} \frac{\pi_l \mathcal{N}(x_n; \mu_l, \Sigma_l)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x_n; \mu_k, \Sigma_k)} + \pi_l \lambda = 0$$

$$\sum_{n=1}^{N} \gamma_{nl} + \lambda \pi_l = 0 \quad \rightarrow \quad \underbrace{\sum_{k=1}^{K} \sum_{n=1}^{N} \gamma_{nl}}_{N} + \lambda \underbrace{\sum_{k=1}^{K} \pi_l}_{1} = 0$$

$$N + \lambda = 0$$

$$\lambda = -N$$

$$\hat{\pi}_l = \frac{\sum_{n=1}^{N} \gamma_{nl}}{N}$$

EM – summary

Step 1: Initialize with some values of the parameters

μ_k, Σ_k, π_k

Step 2: Iterate between

- Computing responsibilities $\gamma_{nk} = \frac{\pi_k \cdot \mathcal{N}(x_n; \mu_k, \Sigma_k)}{\sum_{l=1}^{K} \pi_l \cdot \mathcal{N}(x_n; \mu_l, \Sigma_l))}$
- Computing parameters μ_k, Σ_k, π_k
- Evaluate log-likelihood – check for convergence, and terminate when appropriate

EM -- discussion

- Why is it called Expectation Maximization ?
- In the first part of every iteration, we are computing the **expectation** of Bernoulli RVs:

$$X = \begin{cases} 1 & \text{w.p. } p \\ 0 & \text{w.p. } 1-p \end{cases}$$

$$E[X] = p$$

- In the second part of every iteration, we are **maximizing** the parameters, given the RVs from the previous part.

More on the Expectation step

$P(z_n = k)$

$P(z_n = 1) = 0.7$

$P(z_n = 2) = 0.2$

$P(z_n = 3) = 0.1$

consider z_{nk} as the indicator of cluster membership

$$P(z_{nk}) = \frac{\pi_k \cdot \mathcal{N}(\cdots)}{\sum_{l=1}^{K} \pi_l \cdot \mathcal{N}(\cdots)}$$

EM -- discussion

- Note that each half step improves on the incomplete log likelihood, so we are getting better with each iteration.
  - The proof is more involved and it requires showing that on every half step we improve a particular lower bound on the log likelihood.
  - Picture:

- Again, this property is reminiscent of K-means algorithm.

Connection with K-means

- Let's consider a special case of a mixture model specified as follows:

$$\Sigma_k = \text{diag}(\sigma^2)$$

$$\Sigma_k^{-1} = \text{diag}\left(\frac{1}{\sigma^2}\right)$$

- Recall linear algebra:

$$\det(\Sigma_k) = (\sigma^2)^D$$

- Then, the individual distributions are:

$$P(x_n; \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{D/2} \cdot \sigma^D} \cdot \exp\left(-\frac{1}{2}\frac{1}{\sigma^2}\|x_n - \mu_k\|^2\right)$$

Connection with K-means

- Then, the responsibilities are:

$$\gamma_{nk} = \frac{\pi_k \cdot \exp\left(-\frac{1}{2\sigma^2}\|x_n - \mu_k\|^2\right)}{\sum_k \pi_k \cdot \exp\left(-\frac{1}{2\sigma^2}\|x_n - \mu_k\|^2\right)}$$

- Interpretation: smallest differences in the exponent dominate.

Illustrative example

$K = 3, \quad \pi_1 = \pi_2 = \pi_3 = \frac{1}{3}$

$(x_n - \mu_1) = 1$

$(x_n - \mu_2) = 2$

$(x_n - \mu_3) = 2$

$$\gamma_{n1} = \frac{\pi_1 e^{(-1)^2}}{\pi_1 e^{(-1)^2} + \pi_2 e^{(-2)^2} + \pi_3 e^{(-3)^2}}$$

$$= 0.91$$

Connection with K-means

- Note that the preceding derivation is essentially a principled way of performing "soft" K-means.

$$\frac{\exp(-\beta\|x_n - \mu_\ell\|^2)}{\sum_k \exp(-\beta\|x_n - \mu_k\|^2)}$$

- Can also recover "hard" K-means as a special case:

$\sigma^2 \to 0$

let $\varepsilon = 2\sigma^2$

$$\gamma_{n\ell} = \frac{\pi_\ell \cdot \exp(-\frac{1}{2\sigma^2}\|x_n - \mu_\ell\|^2)}{\sum_k \pi_k \cdot \exp(-\frac{1}{2\sigma^2}\|x_n - \mu_k\|^2)}$$

$$= \frac{\pi_\ell \cdot \exp(-d_\ell/\varepsilon)}{\sum_k \pi_k \cdot \exp(-d_k/\varepsilon)}$$

Connection with K-means

- Can also recover "hard" K-means as a special case:

$$K = 3$$

$$\gamma_{n1} = \frac{\pi_1 e^{-d_1/\varepsilon}}{\pi_1 e^{-d_1/\varepsilon} + \pi_2 e^{-d_2/\varepsilon} + \pi_3 e^{-d_3/\varepsilon}}$$

$$= \frac{\pi_1 e^{-d_1/\varepsilon}}{\pi_1 e^{-d_1/\varepsilon} \left(1 + \frac{\pi_2}{\pi_1} e^{d_1 - d_2/\varepsilon} + \frac{\pi_3}{\pi_1} e^{d_1 - d_3/\varepsilon}\right)}$$

suppose $d_1 < d_2$ $\quad d_2 - d_1 > 0$

$d_1 < d_3$ $\quad d_3 - d_1 > 0$

$\gamma_{n1} \to 1$

$\gamma_{n2} \to 0$

$\gamma_{n3} \to 0$

Discussion

- EM for clustering takes many more steps to converge than K-means.
- Can use K-means to initialize EM for clustering.
- Some issues:
  - In the mixture model, if a mean lands on top of a data point, and cluster variance goes to zero, that term in the likelihood will go to infinity. Avoid this situation.
  - There is also an issue of identifiability, as K! parameters can be permuted.

ECE M146 Introduction to Machine Learning

Lecture 18 - Spring 2021

Prof. Lara Dolecek

ECE Department, UCLA

Today's Lecture

Recap:

- Unsupervised Learning

New topic:

- Ensemble methods: bagging and boosting
- AdaBoost

Recap: Unsupervised learning

- Clustering:
  - K-means for hard membership assignment
  - EM for soft membership assignment

  - Both methods are iterative and they iterate between two steps: cluster membership and parameter refitting.

- PCA for dimensionality reduction
  - Formulate and solve as an optimization problem. Arrive at the dimension specified by the largest eigenvalue. Generalize using SVD.

Today's Lecture

Recap:

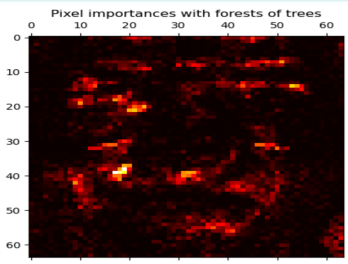
- Unsupervised Learning

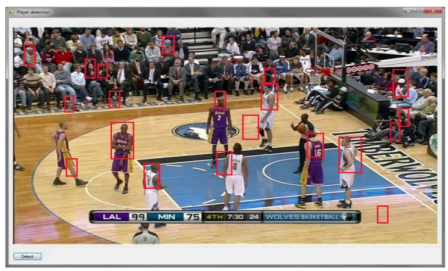
New topic:

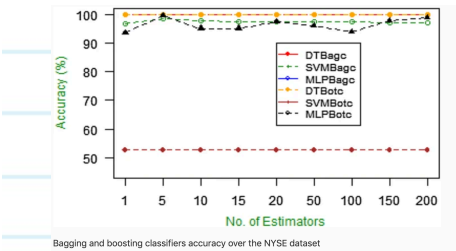
- Ensemble methods: bagging and boosting
- AdaBoost

Applications abound

- Text and image classification: identification of presence/absence of a human in an image; value of each pixel in facial recognition
- Stock market prediction
- Bio informatics: gene to gene interactions, gene expression







https://scikit-learn.org/stable/modules/ensemble.html#adaboost

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00299-5

https://www.semanticscholar.org/paper/Application-of-AdaBoost-Algorithm-in-Basketball-Markoski-Ivankovic/15faf962362505a6062e63511e3e52c9ae92425a

Ensemble methods

- Ensemble of ML algorithms, e.g., classifiers, is a collection of these algorithms whose individual decisions are combined in some way such that the performance of the ensemble is better in the aggregate than that of one of its constituents.

- Let’s consider ensemble of classifiers.

- How to make classifiers non-identical?
- How to combine their results ?

Let's see some choices

- For example, we can add randomness in different initial conditions, and take the average of these classifiers.
- We have touched upon this approach before.

`recall: perceptron is initialization-dependent`

- Other, principled choices are:

1) Bagging (parallel)
2) Boosting (serial)

Bagging

- Bagging stands for **bootstrap aggregation**
- Idea: train K classifiers in parallel, each on re-sampled data, and combine.

Bagging – how

- Suppose we have N training data points. Call this set $\mathcal{D}$.
- Create K new data sets by sampling with replacement from $\mathcal{D}$.
- Call each of these sets $\mathcal{D}_k$, for $1 \leq k \leq K$.

- It is entirely feasible that a given data point appears more than once in one of these sets; it is also entirely feasible that a given data point does not appear in one of these sets.
- What is also feasible is that a data point does not appear in any set!

consequence of sampling

Bagging – intuition

- What bagging does is it suppresses sensitivity to one specific data point, i.e., it reduces the impact of outliers and in turn the variance.

```
result is overlapping effect
```

Example

- Suppose we have 11 binary classifiers, each with individual misclassification rate of 0.3.
- These are relatively weak classifiers.
- We are also going to take an idealistic assumption that they are independent.
- What is the overall misclassification probability, based on the majority vote?

$$\sum_{i=6}^{11} \binom{11}{i} 0.3^i \cdot 0.7^{11-i} \approx 0.07$$

$$\sum_{i=\frac{k+1}{2}}^{k} \binom{k}{i} \varepsilon^i (1-\varepsilon)^{k-i}$$

Upshot

- All we needed was for individual classifiers to have own misclassification rate less than 0.5!
  - Which is, really, just being slightly better than random choice.

weak classifier

- Then, as the number of classifiers increases, the overall misclassification rate decreases to zero.

$$\sum_{i=\frac{k+1}{2}}^{k} \binom{k}{i} \varepsilon^i (1-\varepsilon)^{k-i} \to 0 \quad \text{as} \quad k \to 0 \quad \text{if} \quad \varepsilon < 0.5$$

Boosting

bagging: done in parallel; data sets are generated randomly using sampling *with replacement*

boosting: done serially; data sets are generated opportunistically

- Boosting is a serial approach: base classifiers are trained in a sequential order.
- Each classifier is trained on weighted data, where the weighting factor of a given data point depends on the performance of the preceding classifier on that data point.
- Suppose we have N training data points. Call this set $\mathcal{D}$. In essence, we sequentially create sets $\mathcal{D}_k$, for $1 \leq k \leq K$, where each of these sets is the weighted version of the base set.
- More emphasis is given to more difficult points (i.e., misclassified points).

Today's Lecture

Recap:

- Unsupervised Learning

New topic:

- Ensemble methods: bagging and boosting
- AdaBoost

AdaBoost for Binary Classification

- AdaBoost = Adaptive Boosting.
- Suppose we have N data points x_n, $1 \leq n \leq N$, each with a label t_n in $\{-1,1\}$.
- Suppose we have K binary classifiers.

AdaBoost for Binary Classification

$$\sum_{n=1}^{N} w_n^{(1)} = \sum_{n=1}^{N} \frac{1}{N} = 1$$

Initialization

- Initialize weights $w_n^{(1)} = 1/N$ i.e., each data point is initially given the same importance.

Fit K classifiers

- For $1 \le k \le K$, fit a classifier $y_k(x)$ by minimizing the function J_k

$$J_k = \sum_{n=1}^{N} w_n^{(k)} \, I[y_k(x_n) \neq t_n]$$

Make a prediction

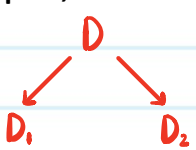
- For a new data point x, compute:

$$\text{sign}\left(\sum_{k=1}^{K} \alpha_k y_k(x)\right)$$

α_k : weight of k^{th} classifier

How to fit an individual classifier ?

- We seek to minimize the function J_k.
- If the classifier is simple, can fit it exhaustively.
- For example, consider a decision tree that makes just one query.



$D_1 \cup D_2 = D$

$D_1 \cap D_2 = \phi$

Update of the weighting coefficients

- Recall that each classifier k has its own weighting coefficients, one per data point, that are derived based on the performance of the preceding classifier.

$0 \leq \epsilon_k \leq 1$

error of k^{th} classifier

$$\epsilon_k = \frac{\sum_{n=1}^{N} w_n^{(k)} I[y_k(x_n) \neq t_n]}{\sum_{n=1}^{N} w_n^{(k)}}$$

- First, capture accuracy:

requirement for later:

$\epsilon_k < 0.5$

- Second, scaling: $\alpha_k = \ln\left(\frac{1-\epsilon_k}{\epsilon_k}\right)$

for $\epsilon_k < 0.5$: $\alpha_k > 0$

Update of the weighting coefficients

- Third, combine to get the weighting coefficients of the next classifier, and renormalize.

1. $w_n^{(k+1)} = w_n^{(k)} \cdot \exp(\alpha_k \cdot I[y_k(x_n) \neq t_n])$

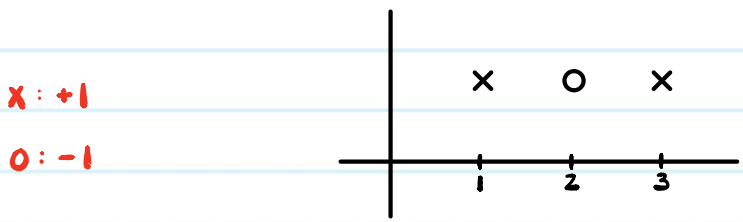
correctly classified: $w_n^{(k+1)} = w_n^{(k)}$

incorrectly classified: $w_n^{(k+1)} = w_n^{(k)} \cdot \exp(\alpha_k)$

2. add up all $w_n^{(k+1)}$ and normalize so their sum is 1

Illustrative example

- Consider the following:



- Clearly no linear classifier alone can perfectly separate this data.
- But, combining simple linear classifiers may work!

Illustrative example

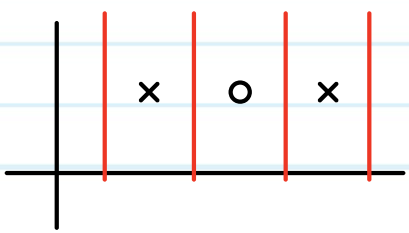
- Step 1: Initialize the weights:

$$w_1^{(3)} = \frac{1}{3} \qquad w_2^{(3)} = \frac{1}{3} \qquad w_3^{(3)} = \frac{1}{3}$$

- Step 2: Fit classifiers sequentially.
- Stage 1: Fit the first classifier to minimize J_1.

$$J_1 = \sum_{n=1}^{3} \frac{1}{3} \cdot I[y_1(x_n) \neq t_n]$$

- What are the choices ?



4 simple classifiers each with a choice of 2 sides, so total of 8 choices

Illustrative example

- Fit the first classifier, ctd.
- At this point, the best we can do is to have one misclassified point.

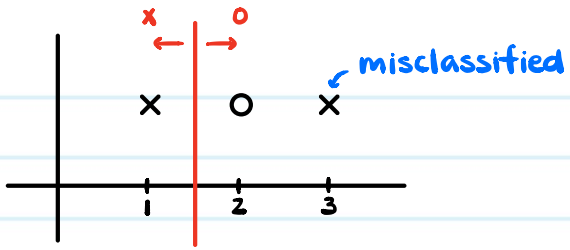
- Compute error (capture accuracy):

$$\varepsilon_1 = \frac{\sum_{n=1}^{3} \frac{1}{3} \cdot \mathbb{1}[y_1(x_n) \neq t_n]}{1} = \frac{1}{3}$$

- Compute scaling:

$$\alpha_1 = \ln\left(\frac{2/3}{1/3}\right)$$
$$= \ln 2$$

Illustrative example



- Stage 1: Fit the first classifier to minimize J_1 (ctd)

- Compute weightings and then renormalize:

$$w_i^{(2)} = w_i^{(1)} \cdot \exp(\alpha_1 \cdot I[y_1(x_i) \neq t_i])$$

- This is how we build the first stage.

$w_1^{(1)} = \frac{1}{3}$

$w_1^{(1)} = \frac{1}{3}$

$w_1^{(1)} = \frac{1}{3} \exp(\ln 2) = \frac{2}{3}$

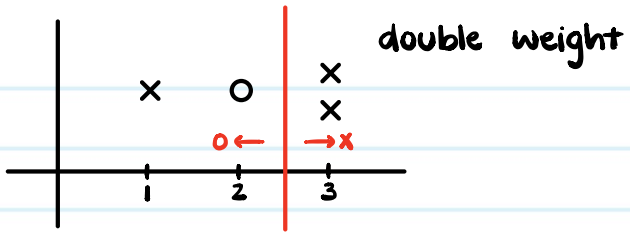
un-normalized

$\frac{1}{3}, \frac{1}{3}, \frac{2}{3}$

normalized

$\frac{1}{4}, \frac{1}{4}, \frac{2}{4}$

Illustrative example



- Stage 2: Fit the second classifier to minimize J_2.

$$J_2 = \sum_{n=1}^{3} w_n^{(2)} \cdot I[y_2(x_n) \neq t_n]$$

- Compute accuracy:

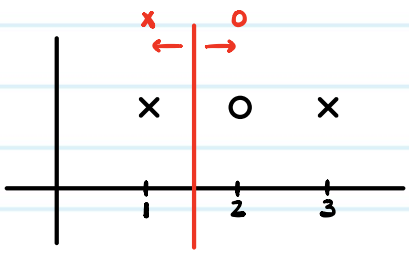
$$\epsilon_2 = \frac{\sum_{n=1}^{3} w_n^{(2)} \cdot I[y_2(x_n) \neq t_n]}{\sum_{n=1}^{3} w_n^{(2)}} = \frac{1}{4}$$

- Compute scaling:

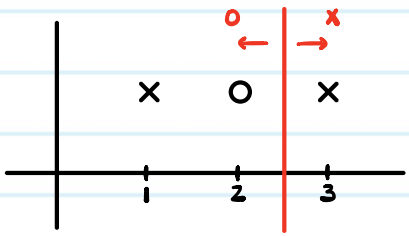
$$\alpha_2 = \ln\left(\frac{3/4}{1/4}\right)$$
$$= \ln 3$$

Illustrative example

- Currently, the overall classifier gives us:



classifier 1



classifier 2

$$\text{sign}(a_1 y_1(x) + a_2 y_2(x))$$

$$= \text{sign}(\ln 2\, y_1(x) + \ln 3\, y_2(x))$$

↑ dominates

Illustrative example

- Stage 2: Fit the second classifier to minimize J_2.

- Compute weightings:

$w_1^{(1)} = \frac{1}{4} \exp(\ln 3) = \frac{3}{4}$

$w_1^{(2)} = \frac{1}{4}$

$w_1^{(3)} = \frac{2}{4}$

un-normalized

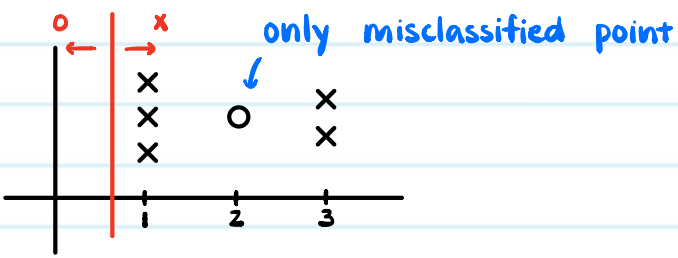
$\frac{3}{4}, \frac{1}{4}, \frac{2}{4}$

normalized

$\frac{1}{2}, \frac{1}{6}, \frac{1}{3}$

Illustrative example

- Stage 3: Fit the third classifier to minimize J_3.

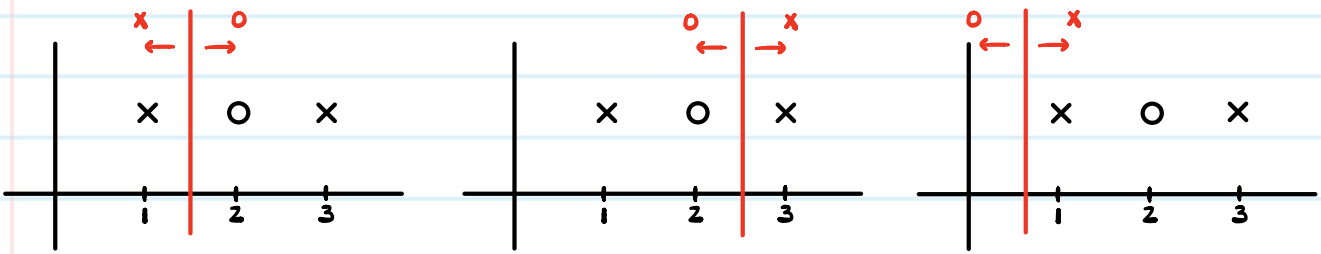


- Compute accuracy: $\varepsilon_3 = \frac{1}{6}$
- Compute scaling:

$$\alpha_2 = \ln\left(\frac{5/6}{1/6}\right)$$
$$= \ln 5$$

Illustrative example

- Currently, the overall classifier gives us:



sign(something)

$$y = \text{sign}\left(\alpha_1 y_1(x) + \alpha_2 y_2(x) + \alpha_3 y_3(x)\right)$$

x : +1
o : −1

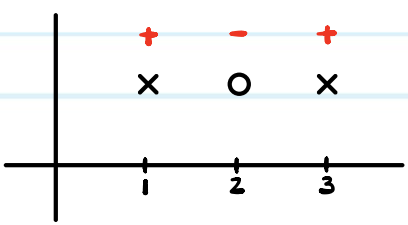
Illustrative example

- Currently, the overall classifier gives us:

1. $\ln 2 - \ln 3 + \ln 5 = \ln 10/3$ +

2. $-\ln 2 - \ln 3 + \ln 5 = \ln 5/6$ −

3. $-\ln 2 + \ln 3 + \ln 5 = \ln 15/2$ +



Comments

- Necessary conditions on the parameters

we had ε_k on each step k

misclassification error

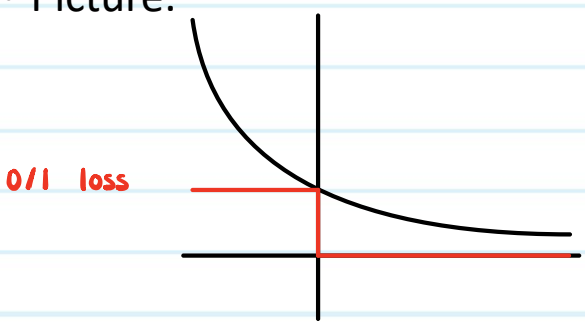
$\varepsilon_k < 0.5$ $\qquad$ $\alpha_k = \ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right) > 0$

Mathematical justification

- The previous algorithm minimizes exponential loss
- Expression:

$$\sum_{n=1}^{N} e^{-t_n \left(\frac{1}{2}\sum_{n=1}^{N} \alpha_k y_k(x_n)\right)}$$

- Picture:



exponential loss as an approximation to 0/1, but is differentiable

Set-up

- Consider labeled data: (x_n, t_n) $t_n = \pm 1$ $1 \le n \le N$
- We minimize this expression:

$$E_m = \sum_{n=1}^{N} e^{-t_n f_m(x_n)}$$

- We keep track of the following quantity:

$$f_m(x_n) = \frac{1}{2} \sum_{l=1}^{m} \alpha_l y_l(x_n)$$

- Can be interpreted as the running weighted average.

Bottom line

- AdaBoost can be interpreted as the minimization of the exponential error in a sequential minimization framework.
- Boosting is a very powerful technique. Boosted Decision Trees are some of the state of the art ML methods for large-scale data.

Summary

- Bagging `parallel procedure: resample data set`
- Boosting `serial procedure: re-weigh the data individually based on the performance of the preceding classifier`

Entropy Connections

- We saw conditional entropy in decision trees, as a rule to decide which attribute to query on.

$$IG(Y, X) = H(Y) - H(Y|X)$$

- We saw cross entropy loss in logistic regression.

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

Entropy Connections

- Entropy itself is a fundamental measure of **lossless** compression.

- Example:

	a	b	c	d	e	f	g	h
p	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$

- Contrast with K-means and PCA which can be used for **lossy** compression.

Example, continued

a	b	c	d	e	f	g	h
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$
0		110		111100		111110	
	10		1110		111101		111111
1	2	3	4	6	6	6	6

avg. length: $\frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{16} \cdot 4 + \frac{4}{64} \cdot 6$

$= 2$ bits

101010 → c6

Example, continued

$$H(X) = -\sum p_i \log_2 p_i$$

$$= \frac{1}{2}\log 2 + \ldots + 4 \cdot \frac{1}{64}\log 64$$

$$= 2 \text{ bits}$$

- Information theory tells us that we cannot compress losslessly below entropy.

If time.

Sequential minimization

- The idea is to minimize E with respect to both scaling coefficients and classifiers. We are going to do this sequentially.
- Mathematical expansion:

Sequential minimization, ctd.

- Math, ctd.

Sequential minimization, ctd.

- Let T_m be the set of indices of points correctly classified by the classifier y_m.

- Let M_m be the set of indices incorrectly classified by the classifier y_m.

Sequential minimization, ctd.

- Decouple the overall error as follows:

Minimization with respect to the last base classifier

Minimization with respect to the scaling parameter

Minimization with respect to the scaling parameter

Minimization with respect to the scaling parameter

More on the scaling coefficients and weights