# Statistical Machine Learning

UoC Stats 37700, Winter quarter

Lecture 8: Ensemble methods, boosting.

- The basic idea of ensemble methods is to learn a family of classifiers $\widehat{f}_1, \ldots, \widehat{f}_K$ , and to output a final decision rule which is a (possibly weighted) vote of these classifiers.
- What are possible reasons for doing so?
  - one uses an “instable” learning procedure and the classifiers are obtained by changing slightly the training set, hoping to make the resulting rule more stable.
  - individual classifiers are “weak” or of limited complexity; but if one manages for different classifiers to concentrate on “subparts” of the learning problem
- Strategies for learning ensembles:
  - Iterative construction of classifiers, using some randomization device or slight changes in the training set.
  - Iterative construction of classifiers, concentrating on examples that have been misclassified up to now.
  - Fixed set of classifiers, choose the weights based on some convex functional to optimize.
- The idea can be applied to regression as well.

- A (possibly weighted) voting procedure for a binary classification problem can be written as

$$\widehat{F}_{ens} = \left( \sum_i \alpha_i \right)^{-1} \sum_{i=1}^{K} \alpha_i \widehat{f}_i \, ,$$

  where the $\widehat{f}_i$ are individual classifiers taking values in $\{-1, 1\}$ and the coefficients $\alpha_i$ are positive.
- The decision is given by the sign of $\widehat{F}_{ens}$.
- Assuming the $\widehat{f}_i$'s belong to a fixed set of "base classifiers" $\mathcal{F} = \{f_t, t \in T\}$, the above can be idealized as

$$\widehat{F}_{ens}(x) = \mathbb{E}_{t \sim \alpha} \left[ f_t(x) \right] \, ,$$

  where $\alpha$ is a distribution on $T$ . Hence the problem of constructing an ensemble method can be seen as one of choosing (depending on the data) a ditribution on the set of base classifiers.

# Bagging

- One of the earliest methods for building an ensemble is Breiman's 'Bagging', initially based on the observation that decision trees were very unstable.
- Strategy: draw bootstrap samples $S_1, S_2, \ldots, S_K$ from $S$ . Use each of them as input for your favorite learning method. Use simple voting of the $K$ resulting classifiers or regressors.
- Provides a sort of "auto-cross-validation" estimate of error (or other quantities) by the "out-of-bag" estimate:
  - For each training example $X_k$, let $i_1, \ldots, i_{n_k}$ the indices of bootstrap samples that do not contain $X_i$ .
  - Predict class or other quantity to estimate for $X_i$ by simple voting over the reduced family $f_{i_1}, \ldots, f_{i_{n_k}}$ .
  - Average over $X_k$ .

# Random Forests (Amit,Geman ’97; Breiman ’01)

- Heuristic randomization strategy for multiple trees that provides some handle on the individual performance/pairwise decorrelation tradeoff.
- Recipe:
  - Construct trees in the usual greedy manner, but at each internal node, instead of considering all possible “questions”, select a random subset of questions to choose from. For example, select a random subset of features.
  - Construct many (a few dozen) random trees this way and perform a regular vote.
  - Not generally needed to prune; a coarse stopping criterion is sufficient.
  - Most important parameters to prune are the ize of the random subset of questions and possibly the stopping criterion,
- Can be combined with bagging and o.o.b. estimates.
- Often surprisingly good in practice in particular for very high-dimensional data.

# The “edge” of a voting procedure

- The “edge” of $\widehat{F}_{ens}$ on instance $(x, y)$ is defined by (assuming $\sum_i \alpha_i = 1$ here)

$$M(\widehat{F}_{ens}, x, y) = y\widehat{F}_{ens} = \sum_i \alpha_i y \widehat{f}_i(x) = \mathbb{E}_{t\sim\alpha}\left[y f_t(x)\right] .$$

- Note the analogy with the margin of a linear classifier; imagine for a moment the $\widehat{f}_i$ are picked from a fixed, finite set $\mathcal{F}$. Then formally, the “feature mapping” of the data maps an instance $x$ to the vector $(f(x), f \in \mathcal{F})$.

# Balancing individual performance and weak pairwise covariance

- Intuitively, if all learnt functions are very close, averaging won't change much.
- On the other hand, if those functions are quite different they are likely to be not very good individually.
- Various heuristics based on this point of view can be used.

# A simple inequality

- (Assume the training set $S$ is fixed for now)
- Assume that $\mu = \mathbb{E}\left[M(\widehat{f}, X, Y)\right] > 0$; then by Chebychev's inequality

$$\mathbb{P}\left[M(\widehat{f}, X, Y) \leq 0\right] \leq \frac{\mathrm{Var}\left[M\right]}{\mu^2}$$

- Note that $\mu$ can be seen as a measure of averaged (over the voting distribution) individual performance of the classifiers:

$$\mu = \mathbb{E}_{X,Y}\left[M(\widehat{f}, X, Y)\right] = \mathbb{E}_{X,Y}\mathbb{E}_t\left[f_t(X)Y\right] = \mathbb{E}_t\mathbb{E}_{X,Y}\left[f_t(X)Y\right]$$

- On the other hand, $\mathrm{Var}\left[M\right]$ can be seen as a measure of averaged pairwise correlation of classifiers drawn from the voting distribution:

$$\mathrm{Var}\left[M\right] = \mathbb{E}_{t,t'\sim\alpha}\left[\mathrm{Cov}_{X,Y}(f_t(X)Y, f_{t'}(X)Y)\right] .$$

# Reinterpretation

- The Chebychev inequality is coarse, but it can be seen to be related to Fisher's discriminant:
- Consider a linear binary classifier $f(x) = w \cdot x - b$ where $b$ is chosen to be the midpoint of the projection of the class centroids $m_1$ and $m_{-1}$ on $w$.
- Then if we define $M_w(x, y) = y f_w(x)$, the same Chebychev's inequality leads to the bound

$$\frac{\mathrm{Var}[M_w]}{\mathbb{E}[M_w]^2} = 4\frac{p_1 s_1 + p_{-1} s_{-1}}{(w \cdot (m_1 - m_{-1}))^2}.$$

- Hence this way we recover (the inverse of) Fisher's discriminant criterion function.

# Relation to game theory

- Consider now a different point of view inspired by the "large margin" interpretation: consider a fixed set of classifiers $\{f_t, t \in T\}$ and assume $T$ to be finite for simplification.
- We can consider the problem of finding the distribution $\alpha$ on $T$ such that the minimum edge on the training examples is maximal:

$$\max_{\alpha} \min_{(X,Y)\in S} M(\alpha, (X, Y)) = \max_{\alpha} \min_{\mathcal{D}} \mathbb{E}_{t\sim\alpha}\mathbb{E}_{(X,Y)\sim\mathcal{D}} [Yf_t(X)] ,$$

  where $\mathcal{D}$ is a (discrete) distribution on the (fixed) points of the sample $S$.
- In game-theoretic point of view, this is the "value" of a zero-sum game where Player 1 ("nature") plays an example $(X, Y)$ at random according to distribution $P$ and Player 2 ("learner") plays a classifier at random according to distribution $\alpha$, and the payoff (for Player 2) is $Yf_t(X)$.

# Summary

- Understanding ensemble methods as a compromise between individual strength of classifiers and weak pairwise covariance (conditional to class) is a second order point of view akin to Fisher's discriminant philosophy.
- In the "large margin" understanding, the goal would be to find the distribution realizing the "mini-max" edge .
- In this case, a crucial difference with the standard geometric large margin view (hard-margin SVM) is the normalization: the 1-norm of the weight vector is normalized to 1 (2-norm in the hard margin SVM).

# The “Boosting” approach

- The Minimax theorem tells us that

$$\max_{\alpha} \min_{\mathcal{D}} \mathbb{E}_{t\sim\alpha}\mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[Yf_t(X)\right]$$
$$= \min_{\mathcal{D}} \max_{\alpha} \mathbb{E}_{t\sim\alpha}\mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[Yf_t(X)\right] = \min_{\mathcal{D}} \max_{f\in\mathcal{F}} \mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[Yf(X)\right] ,$$

- The interpretation of this is the following: the existence of a strictly positive minimax edge $\gamma$ is equivalent to the so-called weak learning property:
  For any distribution $\mathcal{D}$ on the training set, there exists a classifier $f \in \mathcal{F}$ having averaged error less than $\frac{1}{2} - \gamma/2$ under $\mathcal{D}$.

# AdaBoost

- Initially inspired by the weak learning/game theoretic point of view; consists in constructing iteratively a sequence of weighted base classifiers learnt on reweighted versions of the training sample.
- Recipe:
  - (0) Initialize uniform weights $d_{i,0} = 1/n$ on the sample points.
  - (1) Train a classifier $\widehat{f}_k$ based on the weighted sample. Denote $\varepsilon_k$ its weighted error. Stop if $\varepsilon_t = 0$ or $\varepsilon_t \geq 0.5$ .
  - (2) Pick weight for $\widehat{f}_k$ as

$$\alpha_k = \frac{1}{2} \log \frac{1 - \varepsilon_k}{\varepsilon_k} .$$

  - (3) For examples $(X_i, Y_i)$ not correctly classified by $\widehat{f}_k$ , update the corresponding weight by

$$d_{i,k+1} = d_{i,k} \frac{1 - \varepsilon_k}{\varepsilon_k} .$$

  - (4) Renormalize the weights to sum to 1 and reiterate or exit if the number of desired iterations is attained.

# Some examples

We apply boosting to (noiseless) binary classification data using linear classifiers as base classifiers.

- ▷ Example with checkerboard-like classes
- ▷ Example with concentric circles classes

# Properties of AdaBoost

- At step $k+1$, the new reweighting of the sample $S$ is such that classifier $\widehat{f}_k$ has exactly weighted error of 50%.
- The weight $d_{i,k+1}$ of example $i$ at step $k$ is proportional to

$$d_{i,k+1} \propto \exp\left(-\sum_{\ell \leq k} \alpha_\ell \widehat{f}_\ell(X_i) Y_i\right) = \exp\left(-\left(\sum_{i \leq k} \alpha_i\right) \widehat{F}_k(X_i) Y_i\right) ,$$

(where $\widehat{F}_T$ is the ensemble classifier at step $T$) .

# The empirical edge of Adaboost

- Recall the notation $\varepsilon_k$ for the weighted empirical error of classifier $\widehat{f}_k$ . Then the empirical probablity that the edge at step $T$ is less than $\theta$ satifies:

$$\frac{1}{N}\sum_{i=1}^{T}\mathbb{1}\{\widehat{F}_T(X_i)Y_i \leq \theta\} \leq \prod_{i=1}^{T}\sqrt{4\varepsilon_i^{1-\theta}\left(1+\varepsilon_i\right)^{(1+\theta)}}\,.$$

- This implies in particular for the empirical error, putting $\gamma_t = 2(\frac{1}{2} - \varepsilon_k)$:

$$\widehat{\mathcal{E}}(\widehat{F}_T, S) \leq \prod_{i=1}^{T}\sqrt{(1-\gamma_t)(1+\gamma_t)} \leq \exp -\frac{1}{2}\sum_{i\leq T}\gamma_i^2\,.$$

- Furthermore, if for all $t$ , $\gamma_t \geq \gamma_0$ , then the empirical probability that the edge is less that $\theta$ decreases exponentially to 0 for any $\theta \leq \gamma_0/2$ .

# Adaboost as a gradient procedure

- An alternative and fruitful view of AdaBoost is that it realizes a "functional gradient descent" for the exponential loss function over the space of linear combinations classifiers:

$$\ell(G, x, y) = \exp(-G(x)y)\,,\ G = \sum_i \alpha_i f_i,\ ;f_i \in \mathcal{F}\,,$$

- More precisely, if $\widehat{G}_k$ is the current ensemble with unnormalized weights, the learning step in AdaBoost aims at minimizing the "gradient" of the loss function among functional directions $f \in \mathcal{F}$:

$$\text{" } \nabla\widehat{\mathcal{E}}(\ell(\widehat{G}_k), S)\cdot f \text{ "} = \frac{d}{d\alpha}\widehat{\mathcal{E}}(\ell(\widehat{G}_k + \alpha f), S) \propto \sum_{i=1}^{N} d_{i,k}\mathbb{1}\{f(X_i) \neq Y_i\}$$

- Then pick the weight $\alpha_{k+1}$ minimizing the loss function along the chosen direction $\widehat{f}_{k+1}$ .
- AdaBoost is therefore a forward, stagewise additive fitting.

# Multiclass boosting

- The iterative gradient point of view allows to build a natural extension to multiclass with say $L$ classes.
- Consider real-valued classifiers $F(x,y)$ predicting a real value for each class under the constraint $\sum_y F(x,y) = 0$, and the loss function
$$\ell(F,x,y) = \exp(-F(x,y))\,;$$
- Then, let us code the base classifiers by functions $f(x,y)$ taking value 1 on the predicted class and $-(L-1)^{-1}$ otherwise.
- Applying the iterative gradient point of view in this case leads to the following modification of AdaBoost:
  - At step $k+1$, the misclassified examples see their weights multiplied by $(L-1).(1-\varepsilon_k)/\varepsilon_k$
    (remember $\varepsilon_k$ is the weighted error at step $k$).
  - The weright of classifier $k$ is $\alpha_k = \log((L-1).(1-\varepsilon_k)/\varepsilon_k)$.

# Generalizations

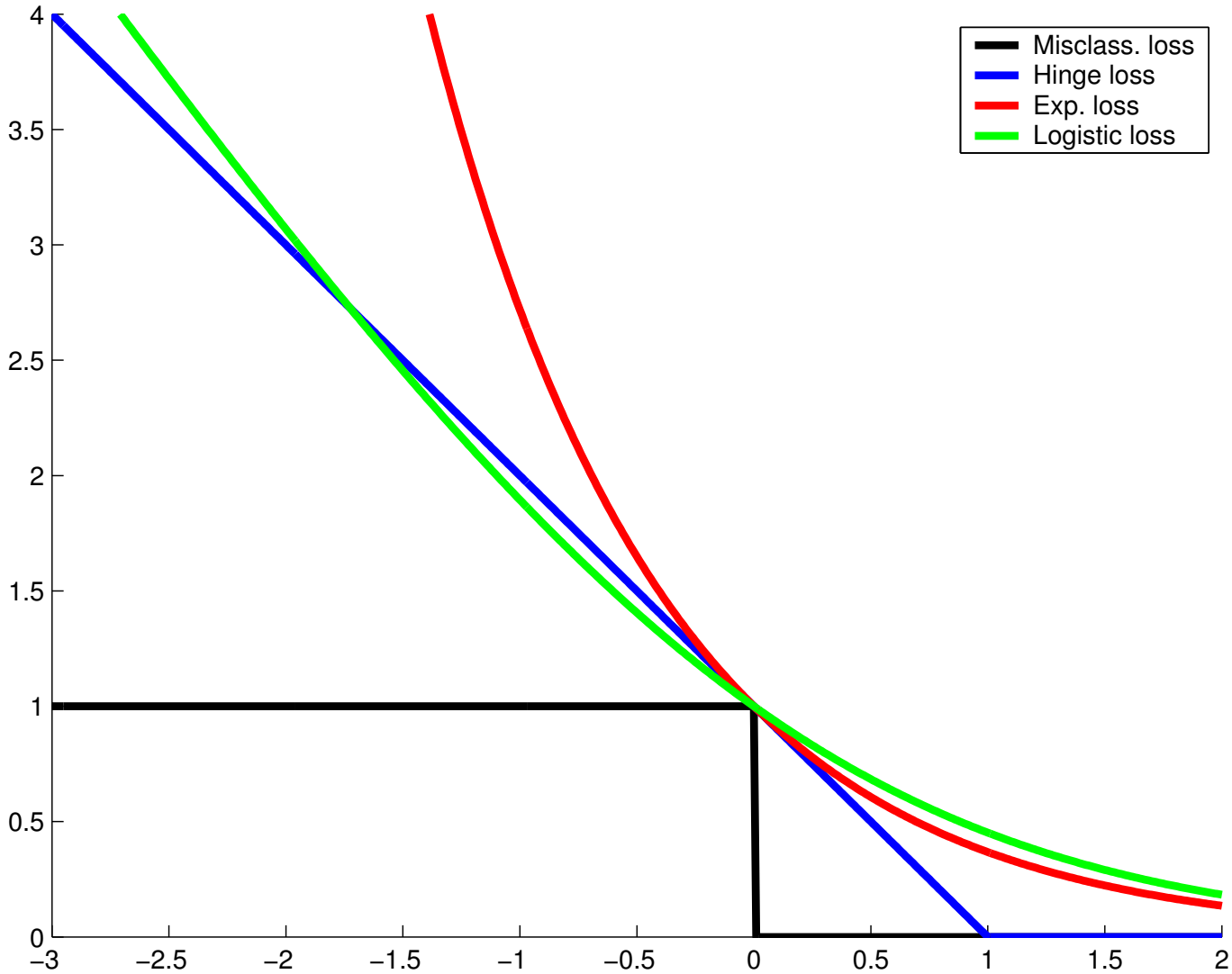
- From the point of view of the minimizing of the exponential loss function, the target function is exactly the log-odds ratio

$$F^*(x) = \frac{1}{2} \log \frac{\mathbb{P}[Y = 1|x]}{\mathbb{P}[Y = -1|x]} .$$

- AdaBoost is thus a method of logistic regression using additive modeling.
- This suggests the possibility of extending this method to other loss functions and/or type of problems. For example, an interesting alternative to the exponential loss (having the same target $F^*$ ) is the minus log-likelihood

$$\ell'(F, x, y) = \log(1 + \exp -2yF(x))$$

- The above loss leads to a more robust procedure wrt. outliers.




Different loss functions of the edge/margin.

# Overfitting and regularized boosting

- AdaBoost works very well in situations where the “base learner” is quite weak, and the data is not too noisy.
- AdaBoost can overfit in noisy situations. This is not too surprising given the agressive nature of the loss minimization procedure which is not traded off with any regularization term.
- One possible form of implicit regularization is to limit the number of iterations.
- Other possibilities are to regularize the objective function, inspired by what is done for the SVM. An example of such a regularized version, assuming we have a finite set $\mathcal{F}$ of base classifiers of cardinality $J$ , is

$$\max_{\gamma,\alpha,\xi} \left( \gamma - \frac{1}{\nu N} \sum_{1 \leq i \leq N} \xi_i \right) ,$$

subject to $Y_i F_\alpha(X_i) \geq \gamma - \xi_i \ , \ \ 1 \leq i \leq n \, ;$
$\xi_j, \alpha_j \geq 0 \, , \ \ 1 \leq j \leq J \, ; \qquad \sum_{j \leq J} \alpha_j = 1 \, .$