STAT 535

12/5/23

Lecture 19

Boosting as G.D.

- Project test labels due Today midnight
- L20 - overview of Clustering

Lecture VII: Combining predictors – Part II

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

November, 2023

Idea of boosting
Ada boost algo
$\uparrow \varphi(f_k) \leq \gamma^k$ $\gamma \in (0,1)$
$\uparrow$ empirical loss
$\downarrow$ boosted f at k
$\varphi(z) = e^{-z}$

Boosting as descent in function space
- Boosted predictors are additive models
- AdaBoost is steepest descent on training set
- A statistical view of boosting

More surrogate losses, more boosting algorithms
- Why the e^{-yf} loss? other surrogate losses
- GradientBoost general φ

Practical remarks and theoretical results
- Practicalities
- [Theoretical results]

Extensions of boosting
- Boosting for multiclass and ranking
- [Multiplicative updates algorithms]

Reading HTF Ch.: 15. Random Forests, 16. Ensembles of predictors, 8. Model inference and averaging, 10. Boosting, Murphy Ch.: 16.3, 16.3.1 Generalized Additive models (ignore the regularization, for "smoother" read "minimize loss function"), 16.4.1-5 [and optionally 8] Bosting, 3.2.4 Bayesian model averaging, 16.6.3, 16.6.1 Ensembles of predictors, Bach Ch.:

AdaBoost Algorithm

AdaBoost Algorithm

Assume $\mathcal{B}$ contains functions b taking values in $[-1, 1]$ or $\{\pm 1\}$

Input M, labeled training set $\mathcal{D}$

Initialize $f = 0$

$w_i^1 = \frac{1}{n}$ weight of datapoint x_i

for $k = 1, 2, \ldots M$

1. "learn classifier for $\mathcal{D}$ with weights w^k" $\Rightarrow$ predictor b^k
2. compute error $\varepsilon^k = \sum_{i=1}^n w_i^k \frac{1 - y_i b^k(x_i)}{2}$
3. set $\beta^k = \frac{1}{2} \ln \frac{1-\varepsilon^k}{\varepsilon^k}$
4. compute new weights $w_i^{k+1} = \frac{1}{Z^k} w_i^k e^{-\beta^k y_i b^k(x_i)}$ where Z^k is the normalization constant that makes $\sum_i w_i^{k+1} = 1$

Output $f(x) = \sum_{k=1}^M \beta^k b^k(x)$

steepest direction

weighted error

optimal step size

$\varepsilon^k \geq \frac{1}{2} - \delta$

$\begin{cases} e^{-\beta^k} & \text{if corr} \\ e^{\beta^k} & \text{if err} \end{cases}$

Normalized

$f^{k+1}(x) = f^k(x) + \beta^k b_k(x)$

constrained gradient descent

Gradient of $\hat{L}_\varphi$

$$\underline{f}^k = \begin{bmatrix} f^k(x^i) \\ \vdots \\ \vdots \end{bmatrix}_{i=1:n} \quad \text{at step } k \qquad y = \begin{bmatrix} y^i \\ \vdots \\ \vdots \end{bmatrix}_{i=1:n}$$

$$b^k = \begin{bmatrix} b^k(x^i) \\ \\ \\ \end{bmatrix}_{i=1:n}$$

$$\hat{L}_\varphi(\underline{f}) = \frac{1}{n} \sum_{i=1}^{n} \varphi\left(y^i \underbrace{f(x^i)}_{f_i,\ i=1:n}\right)$$

$$\varphi(z) = e^{-z}$$

$$\varphi'(z) = -\varphi(z)$$

$$\frac{\partial}{\partial f} \varphi(yf) = -y\varphi(yf)$$

want $f^* \in \mathbb{R}^n$ to minimize $\hat{L}_\varphi$

$f_{(x)} = \sum_k \beta_k b^k(x)$ restricted !!

decrease $\hat{L}_\varphi$ iteratively

$$f^{k+1} = f^k + \beta_k b^k \in \mathbb{R}^n$$

β_k: step size; b^k: direction

$$\hat{L}_\varphi(f) = \frac{1}{n}\sum_{i=1}^{n} \varphi\left(y^i \underbrace{f(x^i)}_{f_i,\ i=1:n}\right)$$

$f \in \mathbb{R}^n$

$$\frac{\partial \hat{L}_\varphi}{\partial f_i} = \frac{1}{n} y^i \varphi'(y^i f_i) = -\frac{1}{n} y^i e^{-y_i f_i}$$

$$\nabla_f \hat{L}_\varphi(f) = \left[\frac{\partial \hat{L}_\varphi(f)}{\partial f_i}\right]$$

f^k

$$\max_{b \in \mathcal{B}} \hat{L}_\varphi(f) - \hat{L}_\varphi(f + \beta b) \approx -\left(\nabla_f \hat{L}_\varphi(f)\right)^T (\beta b) = -\sum_{i=1}^{n} \left(-\frac{1}{n} y^i e^{-y^i f_i}\right)(\beta b_i)$$

$$\propto \sum_i \underbrace{b_i y^i}_{b(x^i)} \underbrace{e^{-y^i f_i}}_{w_i} \quad \leftarrow \max \quad (*)$$

for $b \in \mathcal{B}$

$$\max_{b \in \mathcal{B}} (*) \iff \min_{b \in \mathcal{B}} \sum_i w_i \mathbb{1}_{err(x^i)}$$

↓ Train weak classifier

$b^k \in \mathcal{B}$

⇓

steepest descent direction

Rem $y^i = \pm 1$

if $b(x^i) = \pm 1$

AdaBoost is steepest descent on training set

We will show that boosting is a form of (stochastic) gradient descent on the surrogate loss $\hat{L}_\phi$ (we already know from Part I that AdaBoost pushes $\hat{L}_\phi$ asymptotically towards 0).
Assume we want to minimize the surrogate loss $\hat{L}_\phi$ on the training set. For any finite $\mathcal{D}$, f and $b \in \mathcal{B}$ affect $\hat{L}_\phi$ only via the n-dimensional vectors of their values on $\mathcal{D}$ (which we will abusively denote by f, b)

$$f = \begin{bmatrix} f(x^1) \\ f(x^2) \\ \dots \\ f(x^n) \end{bmatrix} \quad b = \begin{bmatrix} b(x^1) \\ b(x^2) \\ \dots \\ b(x^n) \end{bmatrix} \tag{4}$$

Thus, $\hat{L}_\phi(f)$ is a function of n variables, with partial derivatives

$$\frac{\partial \hat{L}_\phi}{\partial f(x^i)} = \frac{\partial}{\partial f(x^i)} \left[\frac{1}{n} \sum_{i=1}^{n} \phi(y^i f(x^i)) \right] = \frac{1}{n} y^i \phi'(y^i f(x^i)) = -\frac{1}{n} y^i e^{-y^i f(x^i)}, \tag{5}$$

since $\phi'(z) = -e^{-z}$. Imagine a boosting step as trying to find a change βb in f which minimizes the loss $\hat{L}_\phi(f + \beta b)$. This minimization is equivalent to maximizing the decrease in loss $\hat{L}_\phi(f) - \hat{L}_\phi(f + \beta b)$.

The direction of descent

The change in L_ϕ along "direction" b with step size β is approximately

$$\hat{L}_\phi(f) - \hat{L}_\phi(f+\beta b) \approx -\left(\nabla_f \hat{L}_\phi(f)\right)^T(\beta b) = \sum_i \left[\left(\frac{1}{n} y^i e^{-y^i f(x^i)}\right)(\beta b(x^i))\right] \propto \sum_i y^i b(x^i) w_i \quad (6)$$

(denoting/recalling $w_i \propto e^{-y_i f(x^i)}$).
The direction of steepest descent b is therefore the maximizer of

$$\operatorname*{argmax}_{b \in \mathcal{B}} \sum_i w_i y_i b(x^i) \quad (7)$$

where in the sum on the r.h.s we recognize the r of AdaBoost.

- If $b(x^i) = \pm 1$ values, then $1 - y_i b(x^i) = 1_{[i \text{ error}]}$, and maximizing (7) is the same as minimizing the weighted training error $\hat{L}_{01}^w$.
- If b takes real values, then $y_i b(x^i)$ is the margin of example i, and maximizing (7) is a natural objectiv for many training algorithm. Exercise Can you find examples of algorithms/predictors which do/don't maximize the loss in (7)?

More generally (we will use this later), the direction b maximizes

$$\sum_i y_i b(x^i)[-\phi'(y_i f(x^i))] \quad (8)$$

Finding the direction b is equivalent with step 1 of the AdaBoost algorithm, training a weak classifier on the weighted data. The resulting b can be seen as the best approximate of the gradient of L_ϕ in $\mathcal{B}$.

The line minimization = find optimal step size β

f, b = given

Now let us do line minimization: find the optimal step size β in direction b. For this we take the derivative of $\hat{L}_\phi(f + \beta b)$ w.r.t β and set it to 0.

$$\frac{\partial \hat{L}_\phi(f + \beta b)}{\partial \beta} = \sum_i b(x^i)\phi'(y_i f(x^i)) = -\sum_i y_i b(x^i) e^{-y_i f(x^i) - \beta y_i b(x^i)} \tag{9}$$

($f + \beta b$; w_i)

β is the (unique) root of

$$\sum_i w_i y_i b(x^i) e^{-\beta y_i b(x^i)} = 0 \tag{10}$$

If
- $b(x) \in \{-1, 1\}$ then line optimization gives β^k from AdaBoost
- $b(x) \in [-1, 1]$ then line optimization gives β^k from AdaBoost approximately
- $b(x) \in (-\infty, \infty)$ then β amounts to a rescaling of b and is redundant.

$y^i b(x^i) = \begin{cases} 1 & \text{iff correct} \\ -1 & \text{iff error} \end{cases}$

ε_k

$(1 - \varepsilon_k)$ $\left(\sum_{i\,\text{correct}} w_i\right) 1\, e^{-\beta} = \left(\sum_{i\,\text{err}} w_i\right) 1\, e^{\beta} \Rightarrow e^{2\beta} = \frac{1 - \varepsilon_k}{\varepsilon_k}$

Calculating β^k for binary b's

Assume $b(x) \in \{\pm 1\}$.
In this case $y^i b(x^i) = \pm 1$ and we obtain

$$\frac{d\hat{L}_\phi(f + \beta b)}{d\beta} = \sum_{i \text{ corr}} w_i e^{-\beta} - \sum_{i \text{ err}} w_i e^{\beta} = 0 \tag{11}$$

$$0 = (1 - \sum_{i \text{ err}} w_i) - \underbrace{(\sum_{i \text{ err}} w_i)}_{\varepsilon^k} e^{2\beta} \tag{12}$$

$$\beta = \frac{1}{2} \ln \frac{1 - \varepsilon^k}{\varepsilon^k} \tag{13}$$

This is the β^k coefficient of step 4 of AdaBoost

Hence, the AdaBoost algorithm can be seen as minimizing the loss $L_\phi(f)$ by steepest descent in the function space span $\mathcal{B}$.

RealAdaBoost

The third case corresponds to the REALADABOOST in the FHT variant, described here for completeness

REAL ADABOOST ALGORITHM (in the FHT variant)

Assume $\mathcal{B}$ contains real-valued functions

Input M, labeled training set $\mathcal{D}$

Initialize $f = 0$

$w_i^1 = \frac{1}{n}$ weight of datapoint x^i

for $k = 1, 2, \ldots M$

"learn classifier for $\mathcal{D}$ with weights $w^k \Rightarrow b^k$"

compute new weights $w_i^{k+1} = w_i^k e^{-y^i b^k(x^i)}$ and normalize them to sum to 1

Output $f(x) = \sum_{k=1}^{M} b^k(x)$

A statistical view of boosting

It has been shown [?] (FHT) that boosting can also be seen as noisy gradient descent in function space when we replace the finite training set with the true data distribution. The loss function and gradient can be given a probabilistic interpretation. This point of view is useful in two ways:

1. It shows that boosting is asymptotically minimizing a reasonable loss function, so that we can expect the performace/and algorithm behavior on finite samples to be a good predictor on its behaviour with much larger samples.
2. It is an interpretation that allows on to create a very large variety of boosting algorithms, like the LogitBost, Gentle AdaBoost and GradientBoost, presented hereafter.

Assume

- we do boosting "at the distribution level", i.e using P_{XY} instead of the empirical distribution given by $\mathcal{D}$.
- The loss function is $L_\phi(f) = E[e^{-yf(x)}]$.
  The notation $E[]$ denotes expectation w.r.t the joint P_{XY} distribution.
- learning a classifier means "find the best possible minimizer to $L_\phi(f)$"

Is L_ϕ a good loss? 1)

Proposition

Denote $p_x = P_{XY}(y = 1|x)$. The loss $L_\phi(f)$ is minimized by

$$f^*(x) = \frac{1}{2} \ln \frac{P_{XY}(y = 1|x)}{P_{XY}(y = -1|x)} = \frac{1}{2} \ln \frac{p_x}{1 - p_x}$$

And $p_x = \frac{e^{f(x)}}{e^{f(x)} + e^{-f(x)}}$ the logistic function.

Exercise Does the expresion of p_x look familiar? What is the connection?

Proof Since we are minimizing over all possible f's with no restrictions, we can minimize separately for every $f(x)$. Hence, let x be fixed

$$E_{P_{Y|X=x}}[e^{-yf(x)}] = P(y = 1|x)e^{-f(x)} + P(y = -1|x)e^{f(x)}$$

and the gradient is

$$\frac{\partial E[e^{-yf(x)}|x]}{\partial f(x)} = -P(y = 1|x)e^{-f(x)} + P(y = -1|x)e^{f(x)}$$

By setting this to 0 the result follows. □

In summary f^* is the Bayes optimal predictor for L_ϕ. But by the Proposition, f^* is also Bayes optimal for L_{01}. **(Good!)**

Steepest descent on $L_\phi(f)$ is (like) RealAdaboost

Proposition

The Real AdaBoost *(with "learn a classifier" defined at the distribution level) fits an additive logistic regression model f by iterative descent on $L_\phi(f)$.*

Proof The proof is similar to that for the training set case.
Suppose we have a current estimate $f(x)$ and seek to improve it by minimizing $L_\phi(f+b)$ over b. In the proof we assume that b is an arbitrary function, while in practice b will be chosen to best approximate the ideal f within the class $\mathcal{B}$.
Denote by $p_x = P[y=1|x]$ (the true value) and by $\hat{p}_x$ the "estimate"

$$\hat{p}_x = \frac{e^{f(x)}}{e^{f(x)} + e^{-f(x)}} \tag{14}$$

Assume again x is fixed. Then,

$$\begin{aligned} L_\phi(f+b) &= E[e^{-yf(x)-yb(x)}] \\ &= e^{-f(x)}e^{-b(x)}p_x + (1-p_x)e^{f(x)}e^{b(x)} \end{aligned}$$

Taking the derivative and setting it to 0 we obtain the new step:

$$b(x) = \frac{1}{2}\ln\frac{p_x e^{-f(x)}}{(1-p_x)e^{f(x)}} = \frac{1}{2}\left[\ln\frac{p_x}{1-p_x} - \ln\frac{\hat{p}_x}{1-\hat{p}_x}\right] \tag{15}$$

Note that if one could exactly obtain the b prescribed by (15) the iteration would not be necessary.

(Proof, continued)
More interesting than the exact form of b above is the optimization problem that leads to it. Denote $w(x, y) = e^{-yf(x)}$. Then, b is the solution of

$$b = \operatorname*{argmin}_{b \in \mathcal{B}} E_{P_{XY} w(X,Y)}[e^{-Yb}] \tag{16}$$

where $P_{XY} w(X, Y)$ denotes the (unnormalized) **twisted distribution** obtained by multiplying the original data distribution with $w(x, y)$. (Of course, one may have to put some restrictions on P_{XY} and $\mathcal{B}$ in order to obtain a proper distribution.) Finally, note that the new f is $f + b$ and the new weights are $w(x, y)e^{-yb(x)}$ which finishes the proof.
Hence, the Real AdaBoost algorithm can be seen as a form of "noisy gradient" algorithm at the distribution level. (Note that the minimization in equation (16) is over both direction and scale of f.)

What next ?

1. What about L_{01} ?
2. — " — " — φ ← other φ's
3. ✓ convergence $L_\varphi \leq \varepsilon^k$
4. ✓ when stop ?
5. ✓ consistency $f^k \longrightarrow f^*$ Bayes optimal

$n \to \infty$

$k_{(n)} \to \infty$

1) If f not restricted $\Leftrightarrow$ $f^* = \underset{f}{\operatorname{argmin}}\, L_\varphi(f)$ ← Bayes opt for φ

$\Rightarrow f = \underset{f}{\operatorname{argmin}}\, L_{01}(f)$ ← — " — 0-1

Why the e^{-yf} loss? and other L_ϕ losss

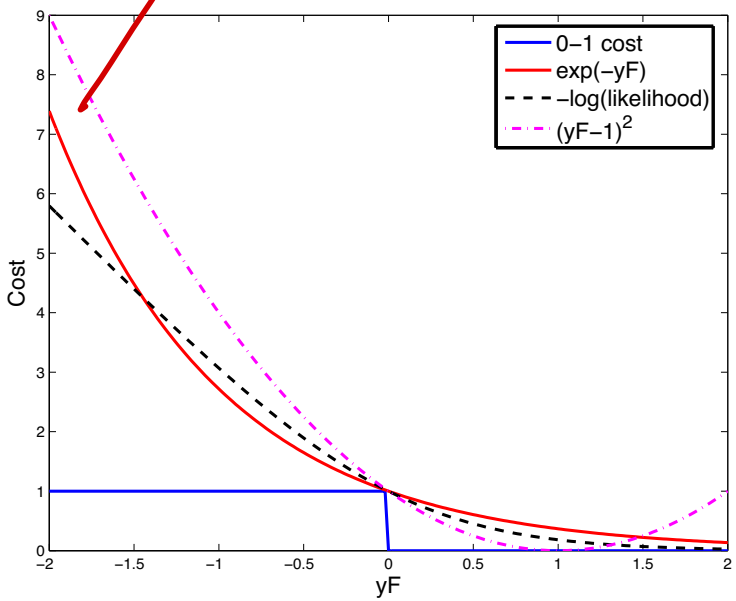
- We saw that L_ϕ is statistically motivated. Now we will see that it is computationally motivated as well.
- Recall: The "true" classification loss L_{01} is nonsmooth (has 0/no derivatives), non-convex. For training, one uses **surrogate losses** of the form $L_\phi(y, f) = \phi(yf)$.
- Want the following properties for ϕ
  - $\phi(z)$ is an upper bound of the 0–1 loss
  - $\phi(z)$ is smooth (has continuous derivatives of any order if f has them); (this lets us use continuous optimization techniques to fit the classifier)
  - $\phi(z)$ is convex (this leads to global optimization, which has been recognized as beneficial in practice; it also allows to prove bounds, rates of convergence and so on)
  - $\phi(z)$ is monothone (decreasing) (thus, when $z > 0$, driving the margins to increase even if the classification is correct).

  These properties are satisfied by $\phi(z) = e^{-z}$

Surrogate losses and boosting algorithms

bad for outliers!

A cornucopia of loss functions



(sometimes good to have $L(z)$ decrease for all $z < 0$, and sometimes bad – causes overfitting)

. . . and of boosting algorithms

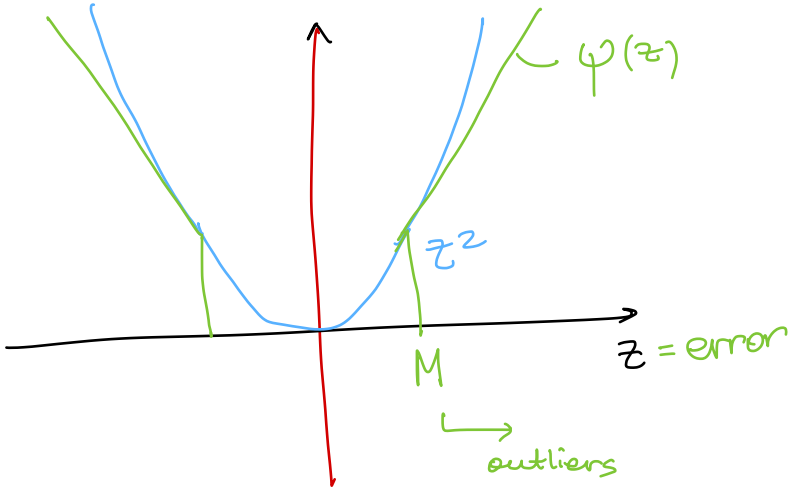
- GentleAdaBoost: approx Newton, $\phi = e^{-z}$
- Least-SquaresBoost: $\phi = (1 - z)^2$ many operations in closed form
- LogitBoost $\phi = \ln(1 + e^{-z})$ slower (almost linear) decrease for $z \ll 0$
- AnyBoost, GradientBoost work with any ϕ

AnyBoost

- choose $\mathcal{B}$
- choose φ
- choose descent method (eg Newton / GD)

$\Rightarrow$ My Boosting Alg

Huber's ψ for regression



M chosen by user

GradientBoost

GradientBoost Algorithm

Given $\mathcal{B}$ contains real-valued functions, loss L_ϕ, ϕ differentiable

Input M, labeled training set $\mathcal{D}$

Initialize $f^0(x) = \beta_0 = \text{argmin}_{\beta \in \mathbb{R}} \hat{L}_\phi(\beta)$

for $k = 0, 1, 2, \ldots M-1$

1. compute $r^i = -y^i \phi'(y^i f(x^i))$ for $i = 1 : n$ } steepest direction
2. fit $b^k(x)$ to data $\{(x^i, r^i)\}$
3. find $\beta_k = \text{argmin}_{\beta \in \mathbb{R}} \hat{L}_\phi(f^k + \beta b^k)$ (univariate optimization) ← step size

update $f^{k+1}(x) = f^k(x) + \beta^k b^k(x)$

Ouput $f^M(x)$

- Can be used for either classification or regression
- Works with any ϕ
- If ϕ convex, step 3 is convex optimization (efficient)
- Proposed first as AnyBoost, later specialized for $\mathcal{B}$ =decision/regression trees, with other tweaks and new name GradientBoost
- When $\mathcal{B}$ =CART
  - step 3 optimizes over every leaf separately
  - depth of trees J represents maximum number of interactions in f; should not be too large ($\mathcal{B}$ must be weak)

with $\mathcal{B}$ = Regression Trees CART with J leaves

Practical aspects

Overfitting in noise When the classes overlap much (many examples in $\mathcal{D}$ hard/impossible to classify correctly) boosting algorithms tend to focus too much on the hard examples, at the expense of overall classification accuracy. The same happens for outliers. Observe also that the loss function(s) in the previous figure, which penalize more as the margin becomes more negative.

Choice of features Often times, the base class $\mathcal{B}$ consists of function of the form $b(x) = x_j - a$, which perform a split on coordinate x_j at point $x_j = a$. They have the advantage that they can be learned and evaluated extremely fast. One can also augment the coordinate vector x with functions of the coordinates (e.g. $x \rightarrow [x_1 \ldots x_d \; x_1x_2 \, x_1x_3 \ldots]$) essentially creating a large set of features, which corresponds to finite but very large $\mathcal{B}$. In such a situation, the number of features d can easily be larger than M the number of b's in the final f. Thus, boosting will be implicitly performing a feature selection task.

When to stop boosting?

$L_{01}(f^k, \mathcal{D}')$ validation loss

The idea of Cross-Validation (CV) is to use an idependent sample from P_{XY}, denoted $\mathcal{D}'$ and called the **validation set** to estimate the expected loss $L_{01}(f)$. When overfitting starts, $L_{01}(f^k)$ will start increasing with k. Boosting is the stopped at the value M that minimizes $\hat{L}_{01}(b^k; \mathcal{D}')$ (denoted L_{cv} below to simplify notation)

AdaBoost with Cross-Validation

Given Training set $\mathcal{D}$ of size n, validation set $\mathcal{D}$ of size n', base classifier $\mathcal{B}$

Initialize

1. while L_{CV} decreases (but for at least 1 step)
   - do a round of boosting on $\mathcal{D}$
   - for $i' = 1 : n'$ compute $f(x^{i'}) \leftarrow f(x^{i'}) + \beta^k b^k(x^{i'})$
   - compute $L_{01}^{CV} = \frac{1}{n'} \sum_{i'} 1_{[y^{i'} f(x^{i'}) < 0]}$

$L_{01}(\mathcal{D}', f^{k+1}) = \frac{1}{n'} \sum_{i \in \mathcal{D}'} L_{01}(f^{k+1}_{(x^i)}, y^i)$

on $\mathcal{D}'$: $f^{k+1}_{(x^i)} = f^k_{(x^i)} + \beta_k b^k(x^i)$ new b^k on $\mathcal{D}'$

How to choose a loss ϕ?

Can we analyze which loss functions ϕ are "better"? Can we offer some guarantees in terms of generalization bounds? The answers are in [?] Bartlett, Jodan & McAuliffe,"Convextity, classification and risk bounds", 2005 (BJM).
We will restrict ourselves to convex, almost everywhere differentiable losss ϕ that are upper bounds of the 0-1 loss.
Let $p = P[Y = 1|X]$, $z = f(X)$. Then the expected loss of classification at X is

$$C_p(z) = p\phi(z) + (1-p)\phi(-z) \tag{17}$$

and the optimal loss is

$$H(p) = \min b^z C_p(z) \quad \text{attained for } f^* = \frac{1}{2} \ln \frac{p}{1-p} \tag{18}$$

Let H^- denote the smallest loss for a misclassification

$$H^-(p) = \inf_{\text{sgn}z = -\text{sgn}(2p-1)} C_p(z) \tag{19}$$

Intuitively, we are minimizing ϕ instead of the "true" misclassification loss, and we want to measure how much we can be off when doing this. The following results say that we can bound the "true" loss $L_{01}(f)$ in terms of the ϕ-loss L_ϕ.
We say ϕ is **classification calibrated** if $H^-(p) > H(p)$ for all $p \neq 1/2$. For ϕ convex, we have that ϕ is classification calibrated iff ϕ differentiable at 0 and $\phi'(0) < 0$.

Proposition

(Theorem 4 in BJM) If ϕ is classification calibrated and convex, then for any classifier F

$$\psi(L_{01}(f) - L_{01}^*) \leq L_\phi(f) - L_\phi^* \tag{20}$$

where L_ϕ^*, L_{01}^* represent respectively the optimal ϕ-loss and optimal classification loss on the given data distribution and ψ is

$$\psi(\theta) \;=\; \phi(0) - H(\frac{1+\theta}{2}) \tag{21}$$

Loss function $\phi(z)$	**Transform function $\psi(\theta)$**
exponential: e^{-z}	$1 - \sqrt{1-\theta^2}$
truncated quadratic: $(\max(1-z,\,0))^2$	θ^2
hinge: $\max(1-z, 0)$	$\lvert\theta\rvert$

In BJM there are also more general theorems that do not assume ϕ is convex.
Furthermore, a convergence rate bound is given, which depends on: the noise in the labels, a complexity parameter of the function class $\mathcal{B}$, the curvature of ϕ. By optimizing this expression w.r.t to $\mathcal{B}$ and ϕ one can theoretically choose the loss function and/or the base classifier.

Consistency of AdaBoost

$\mathcal{F} = \{f's\}$

[?] proved that a large family of boosting algorithms is consistent. Below is an informal version of their main theorem.
Consistency is defined as $L_{01}(f^{t_n}) \to L_{01}^*$ for $n \to \infty$ and a t_n a certain sequence of stopping times that tends to infinity with n.
The conditions for consistency are as follows:

1. The function ϕ is convex, lower bounded (by 0) and calibrated as in section 2
2. The boosting step satisfies a weak leaning condition

$$\hat{L}_\phi(f^k) \leq \gamma \inf_b \hat{L}_\phi(f^{k-1} + \alpha b) + (1-\gamma)\hat{L}_\phi(f^{k-1}) \tag{22}$$

3. The set $\mathcal{B}$ is rich enough that the Bayes loss L_ϕ^* can be attained by convex combinations in $\mathcal{B}$. There is a sequence $\bar{f}^k$ of k terms from $\mathcal{B}$ whose L_ϕ tends to L_ϕ^*
4. The empirical loss $\hat{L}_\phi$ converges to L_ϕ when $n \to \infty$ uniformly over all f which are t_n combinations
5. The empirical risks of $\bar{f}^k$ converge, i.e $\max\{0, |\hat{L}_\phi(\bar{f}^n) - L_\phi(\bar{f}^n)|\} \to 0$, a.s. when $n \to \infty$
6. Algorithmic convergence $\max\{0, |\hat{L}_\phi(f^{t_n}) - \hat{L}_\phi(\bar{f}^n)|\} \to 0$, a.s. when $n \to \infty$. In other words, the boosting algorithm produces an approximate minimizer of the L_ϕ risk if run for t_n iterations
7. $t_n = n^{1-\varepsilon}$ for some $\varepsilon \in (0,1)$ (e.g. increases slowly with n).

M for $|\mathcal{D}| = n$

...and a loss bound

The following result applies to any classifier, but it was developed in response to the idea that "boosting increases the margin" (which we now know is often, but not always true). It proves essentially that large margins counter overfitting. As with all worst-case bounds, the bounds are usually not realistic or practically applicable.

Proposition (to find citation)

Let $\mathcal{F}$ be a model class of VC-dimension h, with $f(x) \in [-1, 1]$ for all x and for all $f \in \mathcal{F}$. Let $\delta > 0$ and $\theta \in (0, 1)$. Then, with probability w.p. $> 1 - \delta$ over training sets

$$L_{01}(f) \leq \frac{1}{n} |\{i \,|\, y^i f(x^i) \leq \theta\}| + \tilde{\mathcal{O}}\left(\sqrt{\frac{h}{n\theta^2}}\right) \tag{23}$$

for any $f \in \mathcal{F}$.

This theorem upper bounds the the true loss $L_{01}(f)$ using the number of margin errors for an arbitrary margin θ. Note that for $\theta = 0$ a margin error is also a classification error, and for $\theta > 0$ the number of margin errors is greater or equal to that of classification errors. Hence, the first term of the bound increases with θ, while the second term decreases. So, if most examples can be classified with a large margin (not necessarily 1), then the bound of Theorem 4 can be tighter.