Boosting

Nuno Vasconcelos

*ECE Department, UCSD*
Classification

- a classification problem has **two types of variables**
  - $X$ - vector of **observations** (features) in the world
  - $Y$ - state (class) of the world

- **e.g.**
  - $x \in X \subseteq \mathbb{R}^2 = \text{(fever, blood pressure)}$
  - $y \in Y = \{\text{disease, no disease}\}$

- $X$, $Y$ related by a (unknown) function

\[
\begin{align*}
  X & \xrightarrow{f(.)} Y = f(X) \\
  x & \mapsto f(x) \quad \forall x
\end{align*}
\]

- **goal:** design a classifier $h: X \rightarrow Y$ such that $h(x) = f(x) \; \forall x$
Linear classifier

- implements the decision rule
  \[ h^*(x) = \begin{cases} 
  1 & \text{if } g(x) > 0 \\
  -1 & \text{if } g(x) < 0 
  \end{cases} = \text{sgn}[g(x)] \]

- for a linearly separable training set \( D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \) we have zero empirical risk when
  \[ y_i(w^T x_i + b) > 0, \ \forall i \]

- and a margin \( \gamma \) when
  \[ y_i(w^T x_i + b) \geq \gamma, \ \forall i \iff \gamma > 0 \]

with \( g(x) = w^T x + b \)
Normalization

- a convenient normalization is to make $|g(x)| = 1$ for the closest point, i.e.

$$\min_{i} \left| w^T x_i + b \right| \equiv 1$$

under which

$$\gamma = \frac{1}{\|w\|}$$

- the SVM is the classifier that maximizes the margin under these constraints

$$\min_{w, b} \|w\|^2 \text{ subjectto } y_i \left( w^T x_i + b \right) \geq 1 \ \forall i$$
The dual problem

- no duality gap, the dual problem is

\[
\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \right\}
\]
subject to \( \sum_i y_i \alpha_i = 0 \)

- once this is solved, the vector

\[
w^* = \sum_i \alpha_i y_i x_i
\]

is the normal to the maximum margin plane

- \( b^* \) can be left a free parameter (false pos. vs misses) or set to

\[
b^* = -\frac{w^T (x^+ + x^-)}{2}
\]
Support vectors

from the KKT conditions, a inactive constraint has zero Lagrange multiplier $\alpha_i$. That is,

- $\alpha_i > 0$ iff $y_i(w^*^T x_i + b^*) = 1$
- i.e. only for points $|w^*^T x_i + b^*| = 1$ which lie at a distance equal to the margin
- these support the plane and are called support vectors
- the decision rule is $f(x) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* x^*_i x + b^* \right]$

the remaining points are irrelevant!
Soft margin optimization

- handles the non-separable case

- instead of solving

\[
\min_{w, b} \|w\|^2 \quad \text{subject to} \quad y_i(w^T x_i + b) \geq 1 \quad \forall i
\]

- we solve the problem

\[
\min_{w, \xi, b} \|w\|^2 \quad \text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i
\]

\[\xi_i \geq 0, \forall i\]

- the \(\xi_i\) are called slacks

- basically, the same as before but points with \(\xi_i > 0\) are allowed to violate the margin
Soft margin optimization

- note that the problem is not really well defined
- by making $\xi_i$ arbitrarily large, any $w$ will do
- we need to penalize large $\xi_i$
- this is leads to the soft margin SVM

$$\min_{w,\xi,b} \|w\|^2 + C \sum_i \xi_i \quad \text{subject to } y_i \left( w^T x_i + b \right) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \forall i$$

- this is called the 1-norm SVM, and is most popular
- we have also discussed the 2-norm case, where penalty is $$f(\xi) = \sum_i \xi_i^2$$
1-norm SVM

the dual problem is

$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \right\}$$

subject to $\sum_i y_i \alpha_i = 0$,

$0 \leq \alpha_i \leq C$

the only difference with respect to the hard margin case is the box constraint on the $\alpha_i$.

this prevents a single outlier from having large impact
Connections to regularization

we talked about penalizing functions that are too complicated, to improve generalization

instead of the empirical risk, we should minimize the regularized risk

\[ R_{\text{reg}}[f] = R_{\text{emp}}[f] + \lambda \Omega[f] \]

the SVM seems to be doing this in some sense:

• it is designed to have as few errors as possible on training set (this is controlled by the soft margin weight \( C \))
• we maximize the margin, by minimizing \( ||w||^2 \) (which is a form of complexity penalty)
• hence, minimizing the margin must be connected to enforcing some form of regularizer
Connections to regularization

- the connection can be made explicit
- consider the 1-norm SVM

\[
\min_{w, \xi, b} \|w\|^2 + C \sum_i \xi_i \quad \text{subject to } y_i g(x_i) \geq 1 - \xi_i \quad \forall i \\
\xi_i \geq 0, \forall i
\]

- the constraints can be rewritten as
  - i) \( \xi_i \geq 0 \) \quad \text{and} \quad ii) \( \xi_i \geq 1 - y_i g(x_i) \)
- which is equivalent to

\[
\xi_i \geq \max[0, 1 - y_i g(x_i)] = [1 - y_i g(x_i)]_+
\]
Connections to regularization

- note that the cost $\|\mathbf{w}\|^2 + C \sum_i \xi_i$ can only increase with larger $\xi_i$
- hence, at the optimal solution

$$\xi_i^* = [1 - y_i g(x_i)]_+$$

- and the problem is

$$\min_{w, b} \|\mathbf{w}\|^2 + C \sum_i [1 - y_i g(x_i)]_+$$

- which is equivalent to

$$\min_{w, b} \sum_i [1 - y_i g(x_i)]_+ + \lambda \|\mathbf{w}\|^2$$

(by making $\lambda = 1/C$)
Connections to regularization

This can be seen as a regularized risk

$$R_{\text{reg}}[f] = \sum_i L[x_i, y_i, f] + \lambda \Omega[f]$$

with

- 1) loss function
  $$L[x, y, g] = [1 - yg(x)]_+$$

- 2) standard regularizer
  $$\Omega[w] = \|w\|^2$$
The SVM loss

it is interesting to compare the SVM loss

\[ L(x, y, g) = [1 - yg(x)]^+ \]

with the “0-1” loss:

- the SVM loss penalizes large negative margins
- assigns some penalty to anything with margin less than 1
- for the “0-1” loss the errors are all the same

the regularizer

- penalizes planes of large w.
- standard measure of complexity in regularization theory
Regularization

- The regularization connection could be used to derive the SVM from classical results, e.g. the representer theorem.

**Theorem:** Let

- $\Omega: [0, \infty) \rightarrow \mathcal{H}$ be a strictly monotonically increasing function,
- $\mathcal{H}$ the RKHS associated with a kernel $k(x, y)$
- $L[y, f(x)]$ a loss function

Then, if

$$f^* = \arg\min_f \left[ \sum_{i=1}^{n} L[y_i, f(x_i)] + \lambda \Omega\left(\|f\|^2\right) \right]$$

- $f^*$ admits a representation of the form

$$f^* = \sum_{i=1}^{n} \alpha_i k(., x_i)$$
Regularization

- in the SVM case, this immediately means that
  \[ w^* = \sum_i \alpha_i y_i k(., x_i) \]
  and we could have derived the SVM from it.

- note that we have seen that it is this theorem which makes
  the problem one of optimization on a finite dim. space
  \[ \alpha^* = \arg\min_{\alpha} \left[ \sum_{i=1}^n L[Y, K\alpha] + \lambda \Omega(\alpha^T K\alpha) \right] \]
even though \( \mathcal{H} \) is infinite dimensional

- in this sense the SVM is really nothing new
  - regularization has been used since the beginning of the century
  - it has just shown that, under appropriate loss, it provides explicit guarantees on generalization error
Boosting

- the regularization connection also allows a **direct link to other techniques**
- one important example is **boosting**
- the basic idea is very different from that of SVMs
  - to **combine many weak classifiers into a powerful committee**
  - a **weak classifier is one that does only slightly better than random**, i.e. slightly less than 50% error
- **key boosting concepts**
  - iterative procedure, **train one weak learner at a time**
  - at each iteration, **focus the attention of the weak learner on the difficult points**, points not well classified so far
  - at the end, **combine all weak predictions into strong prediction**
Boosting

- **Algorithmically** this is done as follows
  1. learn weak classifier
  2. classify all training points and reweight, well classified points get small weights
  3. go to step 1) and repeat M times
  4. combine M weak learners into the “committee classifier”

- in a picture

\[
h(x) = \text{sgn} \left[ \sum_{i=1}^{n} \alpha_i h_i(x) \right]
\]
Fitting additive models

this is an additive model

before getting into details lets see how these are learned

consider basis function expansion of the form

\[
f(x) = \sum_{m=1}^{M} \beta_m b(x, \gamma_m)\]

where

- \(\beta_m\) are the expansion coefficients
- \(b(x, \gamma_m)\) the basis functions of \(x\) parametrized by \(\gamma_m\)

examples:

- polynomial regression: \(b(x, \gamma_m) = x_m\)
- radial basis function approximation: \(b(x, \gamma_m) = G(x, \mu_m, \sigma_m)\)
Fitting additive models

- consider the **problem of learning the expansion from training data**, so as to minimize a loss

\[
\min_{\beta_m, \gamma_m} \sum_{i=1}^{n} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x_i, \gamma_m)\right)
\]

- this is usually a complicated problem

- however, a simple solution can be found when it is possible to **quickly solve the problem of learning just one basis function**

\[
\min_{\beta, \gamma} \sum_{i=1}^{n} L\left(y_i, \beta b(x_i, \gamma)\right)
\]
Forward stagewise additive modeling

- the idea is to **minimize the loss by**
  - sequentially adding new basis functions
  - without adjusting those already added

- **algorithm:**
  - set $f_0(x) = 0$
  - for $m = 1, \ldots, M$
    - let
      \[
      (\beta_m, \gamma_m) = \arg\min_{\beta, \gamma} \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \beta b(x_i, \gamma))
      \]
    - set
      \[
      f_m(x) = f_{m-1}(x) + \beta_m b(x_i, \gamma_m)
      \]

- this is a **greedy optimization procedure**
Boosting

- is forward stagewise additive modeling with
  - $b(x, \gamma_m)$ defined as the weak learners $h_m(x)$
  - the loss function set to
    \[
    L(y, h(x)) = \exp(-yh(x))
    \]
- the cost is
  \[
  \sum_{i=1}^{n} L\left(y_i, \sum_{m=1}^{M} \beta_m h_m(x_i)\right)
  \]
- and the optimal solution at step $m$
  \[
  (\beta_m, h_m) = \arg\min_{\beta, h} \sum_{i=1}^{n} \exp[-y_i(f_{m-1}(x_i) + \beta h(x_i))]
  \]
Boosting

- note that

\[
(\beta_m, h_m) = \arg \min_{\beta, h} \sum_{i=1}^{n} \exp[- y_i (f_{m-1}(x_i) + \beta h(x_i))]
\]

\[
= \arg \min_{\beta, h} \sum_{i=1}^{n} w_i^{(m)} \exp[- y_i \beta h(x_i)]
\]

- where

\[
w_i^{(m)} = \exp[- y_i f_{m-1}(x_i)]
\]

  • does not depend on \(\beta\) or \(h\)
  • can be seen as a weight applied to point \(x_i\)

- note that weight is inversely proportional to exponent of the margin according to the current strong classifier

  • well classified points are less important
The minimization

- is carried out in two steps
  - step 1 is with respect to \( h \)
    - noting that \( y_i, h(x_i) \in \{-1, 1\} \)

\[
\sum_{i=1}^{n} w_i^{(m)} \exp[-y_i \beta h(x_i)] = \sum_{y_i = h(x_i)} w_i^{(m)} e^{-\beta} + \sum_{y_i \neq h(x_i)} w_i^{(m)} e^\beta \\
= e^{-\beta} \sum_{y_i = h(x_i)} w_i^{(m)} + e^\beta \sum_{y_i \neq h(x_i)} w_i^{(m)} \\
= e^{-\beta} \sum_{i} w_i^{(m)} + \left(e^\beta - e^{-\beta}\right) \sum_{y_i \neq h(x_i)} w_i^{(m)} \\
= e^{-\beta} \sum_{i} w_i^{(m)} + \left(e^\beta - e^{-\beta}\right) \sum_{i} w_i^{(m)} I[y_i \neq h(x_i)]
\]

- the first term is constant (wrt \( h \)) and can be dropped
The minimization

- **step 1** reduces to

\[ h_m(x) = \arg \min_h \sum_i w_i^{(m)} I[y_i \neq h(x_i)] \]

i.e. finding the weak learner that achieves the smallest number of errors

- the minimum weighted error of the \(m^{th}\) weak learner is

\[ err_m = \frac{\sum_i w_i^{(m)} I[y_i \neq h_m(x_i)]}{\sum_i w_i^{(m)}} \]

- **step 2** then solves for \(\beta\)
The minimization

**step 2:**

- the cost is
  \[ \sum_{i=1}^{n} w_i^{(m)} \exp[-y_i \beta h(x_i)] = \]
  \[ = e^{-\beta} \sum_i w_i^{(m)} + (e^\beta - e^{-\beta}) \sum_i w_i^{(m)} I[y_i \neq h(x_i)] \]
  \[ = \left( \sum_i w_i^{(m)} \right) \left( e^{-\beta} + (e^\beta - e^{-\beta}) \text{err}_m \right) \]
- and
  \[ \frac{\partial}{\partial \beta} \sum_{i=1}^{n} w_i^{(m)} \exp[-y_i \beta h(x_i)] = \]
  \[ = \left( \sum_i w_i^{(m)} \right) \left( -e^{-\beta} + (e^\beta + e^{-\beta}) \text{err}_m \right) \]
The minimization

- the minimum wrt to $\beta$ satisfies

$$\frac{e^{-\beta}}{e^\beta + e^{-\beta}} = err_m \iff \frac{1}{e^{2\beta} + 1} = err_m \iff e^{2\beta} = \frac{1}{err_m} - 1$$

- and the optimal $\beta$ is

$$\beta_m = \frac{1}{2} \log\left(\frac{1 - err_m}{err_m}\right)$$

- the optimal update is

$$f_m(x) = f_{m-1}(x) + \frac{1}{2} \log\left(\frac{1 - err_m}{err_m}\right) h_m(x)$$
The minimization

weights are updated by

\[ w_i^{(m+1)} = \exp[-y_ i f_m(x_i)] \]
\[ = \exp[-y_ i f_{m-1}(x_i) - y_ i \beta_m h_m(x_i)] \]
\[ = w_i^{(m)} \exp[-y_ i \beta_m h_m(x_i)] \]

noting that

\[ y_ i h_m(x_i) = -2I[y_ i \neq h_m(x_i)] + 1 \]

we have

\[ w_i^{(m+1)} = w_i^{(m)} \exp(2\beta_m I[y_ i \neq h_m(x_i)]) e^{-\beta} \]

and, because we only really care about normalized w’s

\[ w_i^{(m+1)} = w_i^{(m)} \exp(2\beta_m I[y_ i \neq h_m(x_i)]) \]
Adaboost

this leads to the Adaboost algorithm:

1. set $w_i = 1/n$, $i = 1, \ldots, n$

2. for $m = 1, \ldots, M$
   • find the optimal weak classifier using weights $w_i$
     \[
     h_m(x) = \arg \min_h \sum_i w_i^{(m)} I[y_i \neq h(x_i)]
     \]

   • compute
     \[
     \alpha_m = \frac{1}{2} \log \left( \frac{1 - err_m}{err_m} \right)
     \]

   • set
     \[
     w_i^{(m+1)} = w_i^{(m)} \exp(\alpha_m I[y_i \neq h_m(x_i)])
     \]

3. output
   \[
   h(x) = \text{sgn} \left[ \sum_m \alpha_i h_i(x) \right]
   \]
The minimization

\[ \alpha_m = \frac{1}{2} \log \left( \frac{1 - \text{err}_m}{\text{err}_m} \right) \]

\[ \text{err}_m = \frac{\sum w_i^{(m)} I [ y_i \neq h_m(x_i)]}{\sum_i w_i^{(m)}} \]

\[ w_i^{(m+1)} = w_i^{(m)} \exp(\alpha_m I [ y_i \neq h_m(x_i)]) \]

satisfies \( \alpha_m > 0 \) if \( \text{err}_m < 0.5 \)

and from

we have

- correctly classified points: weight stays the same
- errors: weight increases as long as \( \text{err}_m < 0.5 \)
- by keeping \( \text{err}_m \) close to 0.5, the algorithm is less greedy
Weak learners

- should be weak (as in ~50% probability of error)
- popular selection is to threshold individual features
  - assume $x$ is a vector $x = (x^1, \ldots, x^d)^T$
  - to find

$$h_m(x) = \arg\min_h \sum_i w_i^{(m)} I[y_i \neq h(x_i)]$$

- simply cycle through all the features and, for each,
  - find optimal threshold
  - count the errors
  - e.g.

- pick the feature with overall smallest number of errors (right)
Boosting at work

- note that boosting works even the boundaries are quite nonlinear

- e.g.
  - scalar \( x \)
  - Gaussian problem, different \( \sigma \)'s

- iteration 1:
  - all points have same weight
    \[
    h_m(x) = \arg\min_h \sum_i w_i^{(m)} I[y_i \neq h(x_i)]
    \]
  - minimum is 3 errors
Boosting at work

- note that boosting works even the boundaries are quite nonlinear

- e.g.
  - scalar $x$
  - Gaussian problem, different $\sigma$’s

- iteration 1:
  - after weight updates,
    \[
    w_i^{(m)} = \exp[-y_i f_{m-1}(x_i)]
    \]
  - red points get heavier
Boosting at work

- note that boosting works even the boundaries are quite nonlinear

- e.g.
  - scalar $x$
  - Gaussian problem, different $\sigma$’s

- iteration 2:
  - assuming each black error count 1/3,
    \[
    h_m(x) = \arg \min_h \sum_i w_i^{(m)} I[y_i \neq h(x_i)]
    \]
  - minimum error is 1
Boosting at work

- note that boosting works even the boundaries are quite nonlinear
- e.g.
  - scalar x
  - Gaussian problem, different $\sigma$'s
- iteration 2:
  - after weight updates
    \[ w_i^{(m)} = \exp[-y_i f_{m-1}(x_i)] \]
  - some black points get heavier
Boosting at work

► note that boosting works even the boundaries are quite nonlinear

► e.g.
  • scalar $x$
  • Gaussian problem, different $\sigma$’s

► iteration 2:
  • we get the second threshold

$$h_m(x) = \arg \min_h \sum_i w_i^{(m)} I[y_i \neq h(x_i)]$$

• decision rule is something like this
Boosting as feature selection

- we have seen that SVMs are feature selectors
- the same is true for boosting
- at each round
  - select the most discriminant feature (one that best separate the classes)
  - here, \( x \) would be selected first
- note that the feature selection is
  - performed jointly with classifier design
  - explicitly optimal in terms of minimizing classification error
- this is a significant advantage over classical methods
  - select features and then design classifier
Boosting as feature selection

- In fact, boosting is very smart feature selection.

- Feature selection requires:
  - Discrimination
  - Independence

- How can we do this by looking at one feature at a time?

- We do not want copies of the same feature, even if it is discriminant:
  - Think of a problem with 500 features, 300 xs and 200 to ys.
  - Once we picked x, there is no point in picking x again.
  - It would not add anything to our classifier.
  - More generally, we want the features to be as independent as possible.
Boosting as feature selection

- hence
  - there is a tension
  - features correlated with the most discriminant are likely to be discriminant
  - they need to be penalized
  - this is really what the reweighting is accomplishing

- after the first iteration
  - all points well classified along 1st feature are downgraded
  - features correlated with 1st feature will no longer be discriminant
  - all the points left are points where the feature does poorly

- once again, this is done optimally with respect to minimizing classification error!
Boosting as feature selection

in the example

- initially, all points equal weight
- $x$ is most discriminant, picked first
Boosting as feature selection

in the example

• initially, all points equal weight
• $x$ is most discriminant, picked first
• after reweighting (assuming correctly classified points get zero weight)
Boosting as feature selection

in the example

- initially, all points equal weight
- $x$ is most discriminant, picked first
- after reweighting (assuming correctly classified points get zero weight)
- $y$ is now more discriminant, and is picked as second feature
Boosting as feature selection

in the example

• initially, all points equal weight

• x is most discriminant, picked first

• after reweighting (assuming correctly classified points get zero weight)

• y is now more discriminanting, and is picked as second feature

• after reweighting (assuming correctly classified points get zero weight)
Boosting as feature selection

- in the example
  - initially, all points equal weight
  - x is most discriminant, picked first
  - after reweighting (assuming correctly classified points get zero weight)
    - y is now more discriminanting, and is picked as second feature
    - after reweighting (assuming correctly classified points get zero weight)
      - both features are now equally bad, not much more to choose

overall:
  - x is always available and could be picked up again
  - reweighting penalizes the replicas!
Regularization connection

- note that all we did was greedy descent on

\[
\sum_{i=1}^{n} L \left( y_i, \sum_{m=1}^{M} \alpha_m h_m(x_i) \right) = L(y, x) = \exp(-yx)
\]

- since the classifier is

\[
h(x) = \text{sgn} \left[ \sum_{i=1}^{n} \alpha_i h_i(x) \right]
\]

- i.e. defined by hyperplane \( w = (\alpha_1, \ldots, \alpha_M)^T \) on feature space \( z(x) = (h_1(x), \ldots, h_M(x))^T \)

- this is really just the negative exponent of the margin

\[
L(y, x) = \exp(-yg(x))\quad g(x) = w^T x + b
\]
Connections to regularization

- hence, boosting is greedy descent on the loss

\[ L(y, x) = \exp(-yg(x)) \]

- when compared to the SVM loss

\[ L[x, y, g] = [1 - yg(x)]_+ \]

  - similar, also penalizes large negative margins
  - more aggressive penalty of large negatives
  - but some penalty for correct classifications near the margin
  - this can be the source of some problems
  - other versions of boosting try to correct
  - but Adaboost is still the most popular
Connections to regularization

- what about regularization, SRM and all that?
- no explicit regularizer, but an implicit one
- the number M of weak learners
  - without a limit boosting will overfit
  - the limit can be seen as regularizer
  - that limits the number of non-zero components of w
  - (selected features subspace of the feature space of all weak learners)
  - this is really $L_0$ regularization

\[ \Omega[\mathbf{w}] = \| \mathbf{w} \|_0 = \# \{ w_i > 0 \} \]

- in theory, it leads to sparser (more robust) solutions than the $L_2$ regularizer of the SVM!
Any Questions?