On-Line Learning

Nicolò Cesa-Bianchi

Univ. di Milano

SMART
Statistical Multilingual Analysis
for Retrieval and Translation

Summary

Summary

Background

- Theory of repeated games
  (Hannan, 1956; Blackwell, 1956)
- Compression of individual sequences
  (Lempel and Ziv, 1976)
- Gambling and portfolio selection
  (Cover, 1965 and 1991)
- Pattern classification
  (Novikov, 1962; Littlestone, 1989)

Unifying framework

Prediction with expert advice

Binary prediction

- A forecaster predicts a binary sequence one bit at the time

Binary prediction

- A forecaster predicts a binary sequence one bit at the time
- At each step $t = 1, 2, \ldots$ the forecaster predicts the t-th bit knowing the previous $t-1$ bits

$$0100010110\,?\ldots$$

Binary prediction

- A forecaster predicts a binary sequence one bit at the time
- At each step $t = 1, 2, \ldots$ the forecaster predicts the t-th bit knowing the previous $t - 1$ bits

$$0100010110\,?\ldots$$

- After the prediction is made, the t-th bit is observed and the forecaster finds out whether a mistake was made

Binary prediction

- A forecaster predicts a binary sequence one bit at the time
- At each step $t = 1, 2, \ldots$ the forecaster predicts the t-th bit knowing the previous $t - 1$ bits

$$0100010110\,?\ldots$$

- After the prediction is made, the t-th bit is observed and the forecaster finds out whether a mistake was made

Goal

Bound the number of prediction mistakes without making any statistical assumptions on the way the data sequence is generated

The role of experts

- Want a nonstatistical framework where good forecasters can be distinguished from bad forecasters

The role of experts

- Want a nonstatistical framework where good forecasters can be distinguished from bad forecasters
- Any forecaster must use some map of the form

$$\text{past observations} \rightarrow \text{predictions}$$

The role of experts

- Want a nonstatistical framework where good forecasters can be distinguished from bad forecasters
- Any forecaster must use some map of the form

$$\text{past observations} \rightarrow \text{predictions}$$

- For each forecaster, there exists a bit sequence on which a mistake is made at each step

The role of experts

- Want a nonstatistical framework where good forecasters can be distinguished from bad forecasters
- Any forecaster must use some map of the form

$$\text{past observations} \rightarrow \text{predictions}$$

- For each forecaster, there exists a bit sequence on which a mistake is made at each step

Competitive analysis

Compare the performance of the forecaster to that of a set of *reference forecasters* (experts)

A simple example

Forecaster competes against three experts on sequence 1101

A simple example

Forecaster competes against three experts on sequence 1101

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	Mistakes
Expert 1	1	1	1	1	$M_1 = 1$
Expert 2	0	1	1	0	$M_2 = 3$
Expert 3	1	0	1	0	$M_3 = 3$
Forecaster	1	0	1	1	$M = 2$
Bit sequence	1	1	0	1	

A simple example

Forecaster competes against three experts on sequence 1101

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	Mistakes
Expert 1	1	1	1	1	$M_1 = 1$
Expert 2	0	1	1	0	$M_2 = 3$
Expert 3	1	0	1	0	$M_3 = 3$
Forecaster	1	0	1	1	$M = 2$
Bit sequence	1	1	0	1	

Goal (refined)

Predict each sequence almost as well as the best expert for that sequence

A more general prediction model

- Predict an unknown sequence $y_1, y_2, \ldots \in \mathcal{Y}$ (outcome space)

A more general prediction model

- Predict an unknown sequence $y_1, y_2, \ldots \in \mathcal{Y}$ (outcome space)
- Predictions $\widehat{p}$ are chosen from $\mathcal{X}$ (decision space)

A more general prediction model

- Predict an unknown sequence $y_1, y_2, \ldots \in \mathcal{Y}$ (outcome space)
- Predictions $\widehat{p}$ are chosen from $\mathcal{X}$ (decision space)
- Forecasters are scored with their cumulative loss

$$\ell(\widehat{p}_1, y_1) + \ell(\widehat{p}_2, y_2) + \ldots$$

where $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a loss function

A more general prediction model

- Predict an unknown sequence $y_1, y_2, \ldots \in \mathcal{Y}$ (outcome space)
- Predictions $\widehat{p}$ are chosen from $\mathcal{X}$ (decision space)
- Forecasters are scored with their cumulative loss

$$\ell(\widehat{p}_1, y_1) + \ell(\widehat{p}_2, y_2) + \ldots$$

where $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a loss function

Example

- Zero-one loss: $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and $\ell(\widehat{p}, y) = \mathbb{I}_{\{\widehat{p} \neq y\}}$

A more general prediction model

- Predict an unknown sequence $y_1, y_2, \ldots \in \mathcal{Y}$ (outcome space)
- Predictions $\widehat{p}$ are chosen from $\mathcal{X}$ (decision space)
- Forecasters are scored with their cumulative loss

$$\ell(\widehat{p}_1, y_1) + \ell(\widehat{p}_2, y_2) + \ldots$$

where $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a loss function

Example

- Zero-one loss: $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and $\ell(\widehat{p}, y) = \mathbb{I}_{\{\widehat{p} \neq y\}}$
- Quadratic loss: $\mathcal{X} = \mathcal{Y} = [0, 1]$ and $\ell(\widehat{p}, y) = (\widehat{p} - y)^2$

A more general prediction model

- Predict an unknown sequence $y_1, y_2, \ldots \in \mathcal{Y}$ (outcome space)
- Predictions $\widehat{p}$ are chosen from $\mathcal{X}$ (decision space)
- Forecasters are scored with their cumulative loss

$$\ell(\widehat{p}_1, y_1) + \ell(\widehat{p}_2, y_2) + \ldots$$

where $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a loss function

Example

- Zero-one loss: $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and $\ell(\widehat{p}, y) = \mathbb{I}_{\{\widehat{p} \neq y\}}$
- Quadratic loss: $\mathcal{X} = \mathcal{Y} = [0, 1]$ and $\ell(\widehat{p}, y) = (\widehat{p} - y)^2$
- Absolute loss: $\mathcal{X} = [0, 1]$, $\mathcal{Y} = \{0, 1\}$ and $\ell(\widehat{p}, y) = |\widehat{p} - y|$

On-line prediction with expert advice

Measure performance relatively to a set of N experts

At each step $t = 1, 2, \dots$

On-line prediction with expert advice

Measure performance relatively to a set of N experts

At each step $t = 1, 2, \ldots$

1. Get predictions (advice) $f_{1,t}, \ldots, f_{N,t} \in \mathcal{X}$ of the experts

On-line prediction with expert advice

Measure performance relatively to a set of N experts

At each step $t = 1, 2, \dots$

1. Get predictions (advice) $f_{1,t}, \dots, f_{N,t} \in \mathcal{X}$ of the experts
2. Compute prediction $\widehat{p}_t \in \mathcal{X}$

On-line prediction with expert advice

Measure performance relatively to a set of N experts

At each step $t = 1, 2, \dots$

1. Get predictions (advice) $f_{1,t}, \dots, f_{N,t} \in \mathcal{X}$ of the experts
2. Compute prediction $\widehat{p}_t \in \mathcal{X}$
3. Outcome $y_t \in \mathcal{Y}$ is revealed

On-line prediction with expert advice

Measure performance relatively to a set of N experts

At each step $t = 1, 2, \ldots$

1. Get predictions (advice) $f_{1,t}, \ldots, f_{N,t} \in \mathcal{X}$ of the experts
2. Compute prediction $\widehat{p}_t \in \mathcal{X}$
3. Outcome $y_t \in \mathcal{Y}$ is revealed
4. Forecaster incurs loss $\ell(\widehat{p}_t, y_t)$ and each expert i incurs loss $\ell(f_{i,t}, y_t)$

On-line prediction with expert advice

Measure performance relatively to a set of N experts

At each step $t = 1, 2, \dots$

1. Get predictions (advice) $f_{1,t}, \dots, f_{N,t} \in \mathcal{X}$ of the experts
2. Compute prediction $\widehat{p}_t \in \mathcal{X}$
3. Outcome $y_t \in \mathcal{Y}$ is revealed
4. Forecaster incurs loss $\ell(\widehat{p}_t, y_t)$ and each expert i incurs loss $\ell(f_{i,t}, y_t)$

Note

Experts are viewed as abstract entities, generating predictions in an unspecified way

Regret

$$r_{i,t} = \ell(\widehat{p}_t, y_t) - \ell(f_{i,t}, y_t)$$

Regret

$$
\begin{aligned}
r_{i,t} &= \ell(\widehat{p}_t, y_t) - \ell(f_{i,t}, y_t) \\
R_{i,n} &= \sum_{t=1}^{n} r_{i,t} = \sum_{t=1}^{n} \ell(\widehat{p}_t, y_t) - \sum_{t=1}^{n} \ell(f_{i,t}, y_t)
\end{aligned}
$$

Regret

$$
\begin{aligned}
r_{i,t} &= \ell(\widehat{p}_t, y_t) - \ell(f_{i,t}, y_t) \\
R_{i,n} &= \sum_{t=1}^{n} r_{i,t} = \sum_{t=1}^{n} \ell(\widehat{p}_t, y_t) - \sum_{t=1}^{n} \ell(f_{i,t}, y_t)
\end{aligned}
$$

We want to design consistent forecasters, i.e. such that

$$
\lim_{n \to \infty} \frac{1}{n} \left(\max_{i=1,\dots,N} R_{i,n} \right) = 0
$$

for any sequence of outcomes and all choices of expert advice

Weighted average forecasters

- Assume decision space $\mathcal{X}$ is a convex subset of a linear space

Weighted average forecasters

- Assume decision space $\mathcal{X}$ is a convex subset of a linear space
- If $R_{i,t-1}$ is big, then we should predict more like expert i

$$\widehat{p}_t = \frac{\sum_{i=1}^{N} \mu(R_{i,t-1}) f_{i,t}}{\sum_{j=1}^{N} \mu(R_{j,t-1})}$$

where μ is some positive monotone increasing function

Weighted average forecasters

- Assume decision space $\mathcal{X}$ is a convex subset of a linear space
- If $R_{i,t-1}$ is big, then we should predict more like expert i

$$\widehat{p}_t = \frac{\sum_{i=1}^{N} \mu(R_{i,t-1}) f_{i,t}}{\sum_{j=1}^{N} \mu(R_{j,t-1})}$$

  where μ is some positive monotone increasing function
- This is the weighted average forecaster

Potential-based forecasters

- Choose $\mu = \phi'$
  where $\phi : \mathbb{R} \to \mathbb{R}$ is s.t. $\phi, \phi' \geqslant 0$ and ϕ'' exists

Potential-based forecasters

- Choose $\mu = \phi'$
  where $\phi : \mathbb{R} \to \mathbb{R}$ is s.t. $\phi, \phi' \geqslant 0$ and ϕ'' exists
- Weighted average forecaster is then

$$\widehat{p}_t = \frac{\sum_{i=1}^N \phi'(R_{i,t-1}) f_{i,t}}{\sum_{j=1}^N \phi'(R_{j,t-1})}$$

Potential-based forecasters

- Choose $\mu = \phi'$
  where $\phi : \mathbb{R} \to \mathbb{R}$ is s.t. $\phi, \phi' \geqslant 0$ and ϕ'' exists
- Weighted average forecaster is then

$$\widehat{p}_t = \frac{\sum_{i=1}^{N} \phi'(R_{i,t-1}) f_{i,t}}{\sum_{j=1}^{N} \phi'(R_{j,t-1})}$$

Definition

Potential function $\Phi : \mathbb{R}^N \to \mathbb{R}$

$$\Phi(\mathbf{R}) = \psi\left(\sum_{i=1}^{N} \phi(R_i)\right)$$

where $\psi : \mathbb{R} \to \mathbb{R}$ is such that $\psi \geqslant 0, \psi' > 0, \psi'' \leqslant 0$

Blackwell condition

- Using the potential, the prediction at time t gets rewritten as

$$\widehat{p}_t = \frac{\sum_{i=1}^N \nabla\Phi(R_{i,t-1})_i f_{i,t}}{\sum_{j=1}^N \nabla\Phi(R_{j,t-1})_j}$$

Blackwell condition

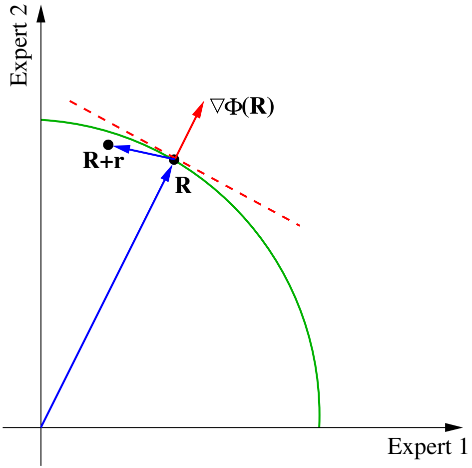
- Using the potential, the prediction at time t gets rewritten as

$$\widehat{p}_t = \frac{\sum_{i=1}^{N} \nabla\Phi(R_{i,t-1})_i f_{i,t}}{\sum_{j=1}^{N} \nabla\Phi(R_{j,t-1})_j}$$

- If the loss is convex, then the following holds

$$\nabla\Phi(\mathbf{R}_{t-1})^\top \mathbf{r}_t \leqslant 0 \qquad \text{(Blackwell condition)}$$

Gradient descent interpretation



Polynomial potential

- Potential function

$$\Phi_p(\mathbf{R}) = \left(\sum_{i=1}^{N} (R_i)_+^p \right)^{2/p} = \|(\mathbf{R})_+\|_p^2 \qquad \text{for } p \geqslant 2$$

Polynomial potential

- Potential function

$$\Phi_p(\mathbf{R}) = \left(\sum_{i=1}^{N} (R_i)_+^p\right)^{2/p} = \|(\mathbf{R})_+\|_p^2 \qquad \text{for } p \geqslant 2$$

- Prediction

$$\widehat{p}_t = \frac{\sum_{i=1}^N \phi'(R_{i,t-1}) f_{i,t}}{\sum_{j=1}^N \phi'(R_{j,t-1})} = \frac{\sum_{i=1}^N (R_{i,t-1})_+^{p-1} f_{i,t}}{\sum_{j=1}^N \left(R_{j,t-1}\right)_+^{p-1}}$$

Exponential potential

- Potential function

$$\Phi_\eta(\mathbf{R}) = \frac{1}{\eta} \ln \left(\sum_{i=1}^{N} e^{\eta R_i} \right) \qquad \text{for } \eta > 0$$

Exponential potential

- Potential function

$$\Phi_\eta(\mathbf{R}) = \frac{1}{\eta} \ln \left(\sum_{i=1}^{N} e^{\eta\, R_i} \right) \qquad \text{for } \eta > 0$$

- Prediction:

$$\widehat{p}_t = \frac{\sum_{i=1}^{N} e^{\eta\left(\widehat{L}_{t-1} - L_{i,t-1}\right)} f_{i,t}}{\sum_{j=1}^{N} e^{\eta\left(\widehat{L}_{t-1} - L_{i,t-1}\right)}} = \frac{\sum_{i=1}^{N} e^{-\eta L_{i,t-1}} f_{i,t}}{\sum_{j=1}^{N} e^{-\eta L_{i,t-1}}}$$

Regret bounds

Loss ℓ is convex and takes values in $[0,1]$

- Polynomial potential with $p = 2 \ln N$

$$\max_{i=1,\dots,N} \frac{R_{i,n}}{n} \leqslant \sqrt{\frac{(2e)}{n} \ln N}$$

Regret bounds

Loss ℓ is convex and takes values in $[0, 1]$

- Polynomial potential with $p = 2 \ln N$

$$\max_{i=1,\dots,N} \frac{R_{i,n}}{n} \leqslant \sqrt{\frac{(2e)}{n} \ln N}$$

- Exponential potential with time-varying parameter η_t

$$\max_{i=1,\dots,N} \frac{R_{i,n}}{n} \leqslant \sqrt{\frac{2}{n} \ln N} + \sqrt{\frac{\ln N}{8n}}$$

Regret bounds

Loss ℓ is convex and takes values in $[0,1]$

- Polynomial potential with $p = 2\ln N$

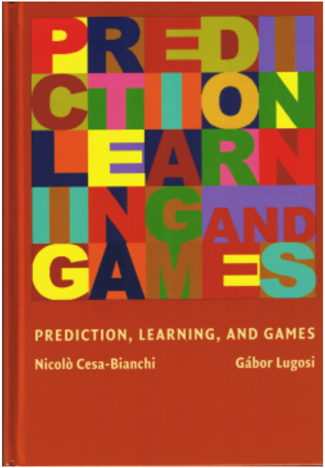
$$\max_{i=1,\dots,N} \frac{R_{i,n}}{n} \leqslant \sqrt{\frac{(2e)}{n}\ln N}$$

- Exponential potential with time-varying parameter η_t

$$\max_{i=1,\dots,N} \frac{R_{i,n}}{n} \leqslant \sqrt{\frac{2}{n}\ln N} + \sqrt{\frac{\ln N}{8n}}$$

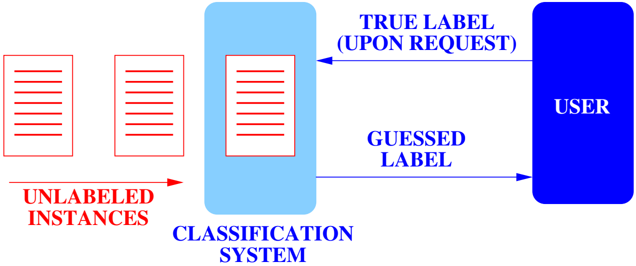
The regret of any forecaster must satisfy:

$$\max_{i=1,\dots,N} \frac{R_{i,n}}{n} = \big(1 - o(1)\big)\sqrt{\frac{2}{n}\ln N}$$


PREDICTION LEARNING AND GAMES
PREDICTION, LEARNING, AND GAMES
Nicolò Cesa-Bianchi
Gábor Lugosi

Summary

On-line classification



Linear classifiers

- Stream of data instances encoded as vectors $x_1, x_2, \dots \in \mathbb{R}^d$

Linear classifiers

- Stream of data instances encoded as vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \in \mathbb{R}^d$
- A binary label $y_t \in \{-1, 1\}$ associated to each $\boldsymbol{x}_t$

Linear classifiers

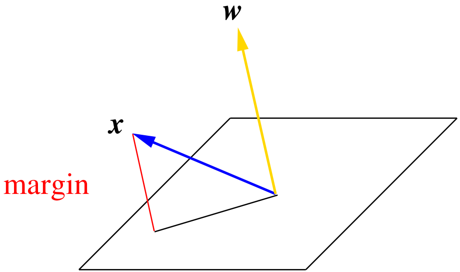
- Stream of data instances encoded as vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \in \mathbb{R}^d$
- A binary label $y_t \in \{-1, 1\}$ associated to each $\boldsymbol{x}_t$
- A linear classifier $\boldsymbol{w}_{t-1} \in \mathbb{R}^d$ predicts label y_t of $\boldsymbol{x}_t$ with

$$\widehat{p}_t = \textsc{sgn}(\boldsymbol{w}_{t-1}^\top \boldsymbol{x}_t) \qquad \boldsymbol{w}_{t-1} \in \mathbb{R}^d$$

Linear classifiers

- Stream of data instances encoded as vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \in \mathbb{R}^d$
- A binary label $y_t \in \{-1, 1\}$ associated to each $\boldsymbol{x}_t$
- A linear classifier $\boldsymbol{w}_{t-1} \in \mathbb{R}^d$ predicts label y_t of $\boldsymbol{x}_t$ with

$$\widehat{p}_t = \operatorname{SGN}(\boldsymbol{w}_{t-1}^\top \boldsymbol{x}_t) \qquad \boldsymbol{w}_{t-1} \in \mathbb{R}^d$$



Linear classifiers (cont.)

If $\widehat{p}_t \neq y_t$ then mistake at step t

Goal

On any arbitrary sequence $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots$ perform not much worse than the best fixed linear classifier

Direct application of experts' framework

One expert for each linear classifier

- Consider the class $\mathcal{F}$ of all linear classifiers $\widehat{p}_t = \textsc{sgn}(\mathbf{u}^\top \mathbf{x}_t)$ for $\mathbf{u} \in \mathbb{R}^d$ with $\|\mathbf{u}\|$ bounded

Direct application of experts' framework

One expert for each linear classifier

- Consider the class $\mathcal{F}$ of all linear classifiers $\widehat{p}_t = \textsc{sgn}(\mathbf{u}^\top \mathbf{x}_t)$ for $\mathbf{u} \in \mathbb{R}^d$ with $\|\mathbf{u}\|$ bounded
- A covering of $\mathcal{F}$ has size exponential in d

Direct application of experts' framework

One expert for each linear classifier

- Consider the class $\mathcal{F}$ of all linear classifiers $\widehat{p}_t = \operatorname{SGN}(\mathbf{u}^\top \mathbf{x}_t)$ for $\mathbf{u} \in \mathbb{R}^d$ with $\|\mathbf{u}\|$ bounded
- A covering of $\mathcal{F}$ has size exponential in d
- Running the weighted average forecaster on the covering requires managing an exponential number of weights

A reduction to prediction with expert advice

One expert for each attribute

- Allocate d experts $F_1, \dots, F_d$

A reduction to prediction with expert advice

One expert for each attribute

- Allocate d experts $F_1, \ldots, F_d$
- On instance $\mathbf{x}_t = (x_{t,1}, \ldots, x_{t,d})$ expert F_j predicts $x_{t,j}$

A reduction to prediction with expert advice

One expert for each attribute

- Allocate d experts $F_1, \ldots, F_d$
- On instance $\boldsymbol{x}_t = (x_{t,1}, \ldots, x_{t,d})$ expert F_j predicts $x_{t,j}$
- Regret $\quad \boldsymbol{r}_t = y_t \, \boldsymbol{x}_t \, \mathbb{I}_{\{\widehat{p}_t \neq y_t\}}$

A reduction (cont.)

- Weighted average forecaster for binary classification

$$\boldsymbol{w}_{t-1} = \nabla\Phi(\boldsymbol{R}_{t-1}) \qquad \widehat{p}_t = \mathrm{SGN}(\boldsymbol{w}_{t-1}^\top \boldsymbol{x}_t)$$

A reduction (cont.)

- Weighted average forecaster for binary classification

$$\boldsymbol{w}_{t-1} = \nabla\Phi(\boldsymbol{R}_{t-1}) \qquad \widehat{p}_t = \mathrm{SGN}(\boldsymbol{w}_{t-1}^\top \boldsymbol{x}_t)$$

- We need Blackwell condition $\quad \boldsymbol{w}_{t-1}^\top \boldsymbol{r}_t \leqslant 0 \quad$ to hold

A reduction (cont.)

- Weighted average forecaster for binary classification

$$w_{t-1} = \nabla\Phi(R_{t-1}) \qquad \widehat{p}_t = \mathrm{SGN}(w_{t-1}^\top x_t)$$

- We need Blackwell condition $\quad w_{t-1}^\top r_t \leqslant 0 \quad$ to hold
- Indeed,

$$w_{t-1}^\top r_t = y_t\, w_{t-1}^\top x_t\, \mathbb{I}_{\{\widehat{p}_t \neq y_t\}} = \begin{cases} 0 & \text{if } \mathbb{I}_{\{\widehat{p}_t \neq y_t\}} = 0 \\ < 0 & \text{otherwise} \end{cases}$$

since $\mathbb{I}_{\{\widehat{p}_t \neq y_t\}} = 1$ iff $\mathrm{SGN}(w_{t-1}^\top x_t) \neq y_t$

A reduction (cont.)

- Weighted average forecaster for binary classification

$$\boldsymbol{w}_{t-1} = \nabla\Phi(\boldsymbol{R}_{t-1}) \qquad \widehat{p}_t = \textsc{sgn}(\boldsymbol{w}_{t-1}^\top \boldsymbol{x}_t)$$

- We need Blackwell condition $\quad \boldsymbol{w}_{t-1}^\top \boldsymbol{r}_t \leqslant 0 \quad$ to hold
- Indeed,

$$\boldsymbol{w}_{t-1}^\top \boldsymbol{r}_t = y_t\, \boldsymbol{w}_{t-1}^\top \boldsymbol{x}_t\, \mathbb{I}_{\{\widehat{p}_t \neq y_t\}} = \begin{cases} 0 & \text{if } \mathbb{I}_{\{\widehat{p}_t \neq y_t\}} = 0 \\ < 0 & \text{otherwise} \end{cases}$$

  since $\mathbb{I}_{\{\widehat{p}_t \neq y_t\}} = 1$ iff $\textsc{sgn}(\boldsymbol{w}_{t-1}^\top \boldsymbol{x}_t) \neq y_t$
- The potential-based analysis can be adapted to bound the regret against any fixed linear classifier

Formulation as an incremental algorithm

We want to express $\boldsymbol{w}_t = \nabla\Phi(\boldsymbol{R}_t)$ recursively as $\boldsymbol{w}_t = F(\boldsymbol{w}_{t-1})$

Formulation as an incremental algorithm

We want to express $\boldsymbol{w}_t = \nabla\Phi(\boldsymbol{R}_t)$ recursively as $\boldsymbol{w}_t = F(\boldsymbol{w}_{t-1})$

Definition

A potential $\Phi : \mathbb{R}^d \to \mathbb{R}$ is Legendre if Φ is strictly convex, differentiable, and has a convex domain (and . . .)

Formulation as an incremental algorithm

We want to express $\boldsymbol{w}_t = \nabla\Phi(\mathbf{R}_t)$ recursively as $\boldsymbol{w}_t = F(\boldsymbol{w}_{t-1})$

Definition

A potential $\Phi : \mathbb{R}^d \to \mathbb{R}$ is Legendre if Φ is strictly convex, differentiable, and has a convex domain (and ...)

If a potential is Legendre, then $\nabla\Phi$ is invertible

$$\boldsymbol{w}_t = \nabla\Phi(\mathbf{R}_t) = \nabla\Phi(\mathbf{R}_{t-1} + \boldsymbol{r}_t) = \nabla\Phi\Big((\nabla\Phi)^{-1}(\boldsymbol{w}_{t-1}) + \boldsymbol{r}_t\Big)$$

Formulation as an incremental algorithm

We want to express $\boldsymbol{w}_t = \nabla\Phi(\boldsymbol{R}_t)$ recursively as $\boldsymbol{w}_t = F(\boldsymbol{w}_{t-1})$

Definition

A potential $\Phi : \mathbb{R}^d \to \mathbb{R}$ is Legendre if Φ is strictly convex, differentiable, and has a convex domain (and ...)

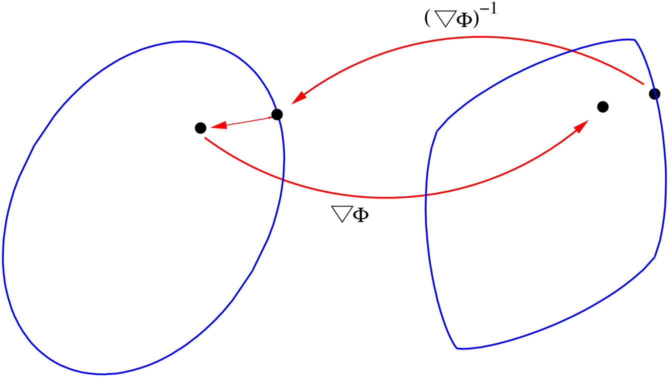
If a potential is Legendre, then $\nabla\Phi$ is invertible

$$\boldsymbol{w}_t = \nabla\Phi(\boldsymbol{R}_t) = \nabla\Phi(\boldsymbol{R}_{t-1} + \boldsymbol{r}_t) = \nabla\Phi\Big((\nabla\Phi)^{-1}(\boldsymbol{w}_{t-1}) + \boldsymbol{r}_t\Big)$$

Update rule

$$\boldsymbol{w}_t = \nabla\Phi\Big((\nabla\Phi)^{-1}(\boldsymbol{w}_{t-1}) + y_t\,\boldsymbol{x}_t\,\mathbb{I}_{\{\widehat{p}_t \neq y_t\}}\Big)$$

Incremental formulation (cont.)



$$w_t = \nabla\Phi\Big((\nabla\Phi)^{-1}(w_{t-1}) + y_t\, x_t\, \mathbb{I}_{\{\widehat{p}_t \neq y_t\}} \Big)$$

Application to polynomial potential

- Polynomial potential $\Phi_p(\cdot) = \|\cdot\|_p^2$ is Legendre

$$\left(\nabla \tfrac{1}{2}\|\mathbf{u}\|_p^2\right)_i = \frac{\mathrm{SGN}(u_i)\,|u_i|^{p-1}}{\|\mathbf{u}\|_p^{p-2}} \qquad \left(\nabla \tfrac{1}{2}\|\mathbf{u}\|_p^2\right)^{-1} = \nabla \tfrac{1}{2}\|\mathbf{u}\|_q^2$$

where q is such that $1/p + 1/q = 1$

Application to polynomial potential

- Polynomial potential $\Phi_p(\cdot) = \|\cdot\|_p^2$ is Legendre

$$\left(\nabla \tfrac{1}{2}\|\mathbf{u}\|_p^2\right)_i = \frac{\textsc{sgn}(u_i)\,|u_i|^{p-1}}{\|\mathbf{u}\|_p^{p-2}} \qquad \left(\nabla \tfrac{1}{2}\|\mathbf{u}\|_p^2\right)^{-1} = \nabla \tfrac{1}{2}\|\mathbf{u}\|_q^2$$

where q is such that $1/p + 1/q = 1$

- When $p = 2$ we have $\nabla\Phi_2(\mathbf{R}) = \mathbf{R}$

Application to polynomial potential

- Polynomial potential $\Phi_p(\cdot) = \|\cdot\|_p^2$ is Legendre

$$\left(\nabla \tfrac{1}{2}\|\mathbf{u}\|_p^2\right)_i = \frac{\mathrm{SGN}(u_i)\,|u_i|^{p-1}}{\|\mathbf{u}\|_p^{p-2}} \qquad \left(\nabla \tfrac{1}{2}\|\mathbf{u}\|_p^2\right)^{-1} = \nabla \tfrac{1}{2}\|\mathbf{u}\|_q^2$$

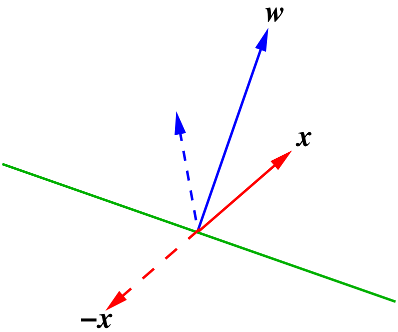
where q is such that $1/p + 1/q = 1$

- When $p = 2$ we have $\nabla\Phi_2(\mathbf{R}) = \mathbf{R}$
- The update rule then is simply

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + y_t\,\boldsymbol{x}_t\,\mathbb{I}_{\{\widehat{p}_t \neq y_t\}}$$

the Perceptron algorithm (Rosenblatt, 1952)

The Perceptron algorithm



$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + y_t \, \boldsymbol{x}_t \, \mathbb{I}_{\{\widehat{p}_t \neq y_t\}}$$

Application to the exponential potential

- The exponential potential $\Phi_{\exp}(\mathbf{R}) = e^{R_1} + \cdots + e^{R_d}$ is Legendre

Application to the exponential potential

- The exponential potential $\Phi_{\exp}(\mathbf{R}) = e^{R_1} + \cdots + e^{R_d}$ is Legendre
- The update rule is

$$w'_{i,t} = w'_{i,t-1}\, e^{\eta\, r_{i,t-1}}$$

Application to the exponential potential

- The exponential potential $\Phi_{\exp}(\mathbf{R}) = e^{R_1} + \cdots + e^{R_d}$ is Legendre
- The update rule is

$$
\begin{aligned}
w'_{i,t} &= w'_{i,t-1}\, e^{\eta\, r_{i,t-1}} \\
w_{i,t} &= \frac{w'_{i,t}}{\displaystyle\sum_{k=1}^{d} w'_{k,t}}
\end{aligned}
$$

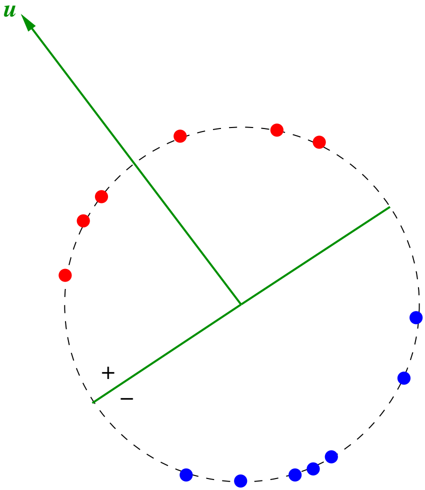
Application to the exponential potential

- The exponential potential $\Phi_{\exp}(\mathbf{R}) = e^{R_1} + \cdots + e^{R_d}$ is Legendre
- The update rule is

$$
\begin{aligned}
w'_{i,t} &= w'_{i,t-1}\, e^{\eta\, r_{i,t-1}} \\
w_{i,t} &= \frac{w'_{i,t}}{\displaystyle\sum_{k=1}^{d} w'_{k,t}}
\end{aligned}
$$

- This is the Winnow algorithm (Littlestone, 1988)

The linearly separable case



Comparison between poly. and exp. potential

Mistake bounds for linearly separable sequences

Comparison between poly. and exp. potential

Mistake bounds for linearly separable sequences

$$\frac{p-1}{2}\boxed{\left(X_p\,\|\mathbf{u}\|_q\right)^2}$$

poly. potential

q is such that $1/p + 1/q = 1$

Comparison between poly. and exp. potential

Mistake bounds for linearly separable sequences

$$\frac{p-1}{2}\boxed{\left(X_p \|\mathbf{u}\|_q\right)^2} \qquad \text{poly. potential}$$

$$\big(1+o(1)\big)\ln(2d)\boxed{\left(X_\infty \|\mathbf{u}\|_1\right)^2} \qquad \text{exp. potential}$$

q is such that $1/p + 1/q = 1$

Comparison between poly. and exp. potential

Mistake bounds for linearly separable sequences

$$\frac{p-1}{2}\left(X_p\,\|\mathbf{u}\|_q\right)^2 \qquad \text{poly. potential}$$

$$\big(1+o(1)\big)\ln(2d)\,\left(X_\infty\,\|\mathbf{u}\|_1\right)^2 \qquad \text{exp. potential}$$

q is such that $1/p + 1/q = 1$

- Bound for exp. potential assumes tuning (previous knowledge of X_∞ and choice of $\|\mathbf{u}\|_1$)

Comparison between poly. and exp. potential

Mistake bounds for linearly separable sequences

$$\frac{p-1}{2}\left(X_p \|\mathbf{u}\|_q\right)^2 \qquad \text{poly. potential}$$

$$\big(1+o(1)\big)\ln(2d)\,(X_\infty \|\mathbf{u}\|_1)^2 \qquad \text{exp. potential}$$

q is such that $1/p + 1/q = 1$

- Bound for exp. potential assumes tuning (previous knowledge of X_∞ and choice of $\|\mathbf{u}\|_1$)
- Both bounds depend on pairs of dual norms: $\|\mathbf{x}\|_p \|\mathbf{u}\|_q$ vs. $\|\mathbf{x}\|_\infty \|\mathbf{u}\|_1$

Comparison between poly. and exp. potential

Mistake bounds for linearly separable sequences

$$\frac{p-1}{2}\boxed{\left(X_p \|\mathbf{u}\|_q\right)^2} \qquad \text{poly. potential}$$

$$(1+o(1))\ln(2d)\boxed{\left(X_\infty \|\mathbf{u}\|_1\right)^2} \qquad \text{exp. potential}$$

q is such that $1/p + 1/q = 1$

- Bound for exp. potential assumes tuning (previous knowledge of X_∞ and choice of $\|\mathbf{u}\|_1$)
- Both bounds depend on pairs of dual norms: $\|\mathbf{x}\|_p \|\mathbf{u}\|_q$ vs. $\|\mathbf{x}\|_\infty \|\mathbf{u}\|_1$
- For $p \approx 2\ln d$ the bounds are essentially equal

Comparison for spherical potential

- Consider a sequence $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \dots$ such that $\boldsymbol{x}_t \in \{-1, 1\}^d$ and $y_t = \text{SGN}(x_{1,t})$

Comparison for spherical potential

- Consider a sequence $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots$ such that $\mathbf{x}_t \in \{-1, 1\}^d$ and $y_t = \textsc{sgn}(x_{1,t})$
- Then $\mathbf{u} = (1, 0, \dots, 0)$ is an optimal classifier (no loss)

Comparison for spherical potential

- Consider a sequence $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \dots$ such that $\boldsymbol{x}_t \in \{-1,1\}^d$ and $y_t = \text{SGN}(x_{1,t})$
- Then $\boldsymbol{u} = (1,0,\dots,0)$ is an optimal classifier (no loss)
- Moreover,

$$\left(\|\boldsymbol{u}\|_2 X_2\right)^2 = d \qquad \text{and} \qquad \left(\|\boldsymbol{u}\|_1 X_\infty\right)^2 = 1$$

Comparison for spherical potential

- Consider a sequence $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \ldots$ such that $\boldsymbol{x}_t \in \{-1, 1\}^d$ and $y_t = \text{SGN}(x_{1,t})$
- Then $\mathbf{u} = (1, 0, \ldots, 0)$ is an optimal classifier (no loss)
- Moreover,

$$\left(\|\mathbf{u}\|_2 X_2\right)^2 = d \qquad \text{and} \qquad \left(\|\mathbf{u}\|_1 X_\infty\right)^2 = 1$$

- Then mistake bounds are

$$\begin{array}{cl} d & \text{polynomial potential, } p = 2 \\ 4\ln(2d) & \text{exponential potential} \end{array}$$

an exponential advantage (verified by experiments)

Comparison for spherical potential

- Consider a sequence $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots$ such that $\mathbf{x}_t \in \{-1, 1\}^d$ and $y_t = \text{SGN}(x_{1,t})$
- Then $\mathbf{u} = (1, 0, \dots, 0)$ is an optimal classifier (no loss)
- Moreover,

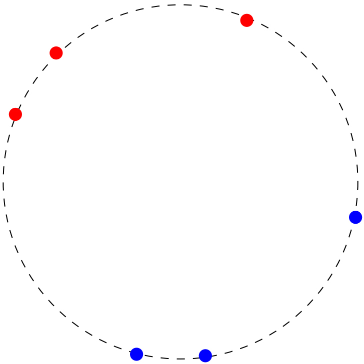
$$\left(\|\mathbf{u}\|_2 X_2\right)^2 = d \qquad \text{and} \qquad \left(\|\mathbf{u}\|_1 X_\infty\right)^2 = 1$$

- Then mistake bounds are

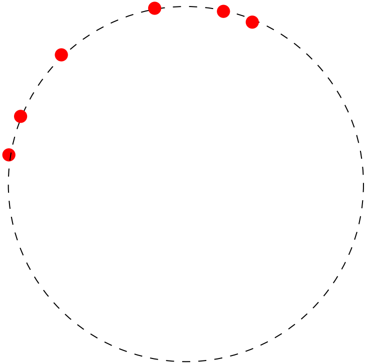
$$\begin{array}{rl} d & \text{polynomial potential, } p = 2 \\ 4\ln(2d) & \text{exponential potential} \end{array}$$

  an exponential advantage (verified by experiments)
- Opposite situation when instances $\mathbf{x}_t$ are sparse and best expert $\mathbf{u}$ is dense

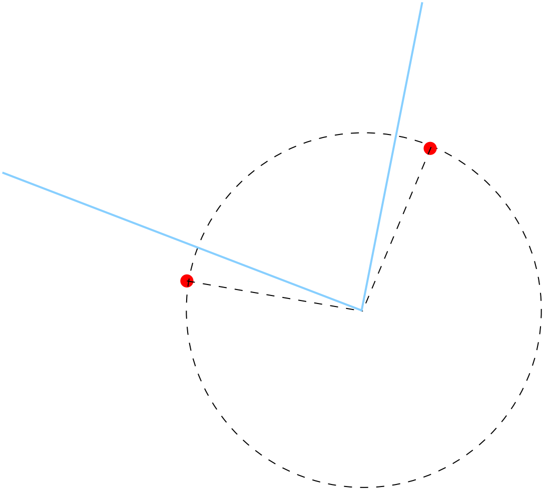
The cone of consistent hyperplanes



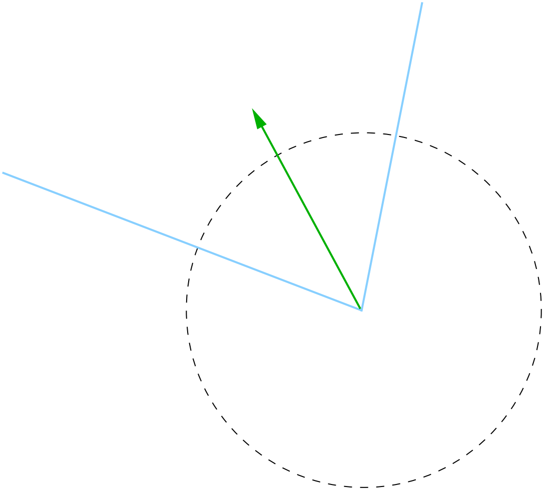
The cone of consistent hyperplanes



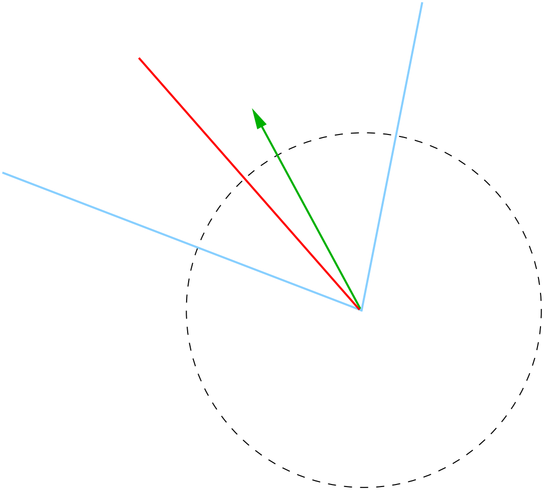
The cone of consistent hyperplanes



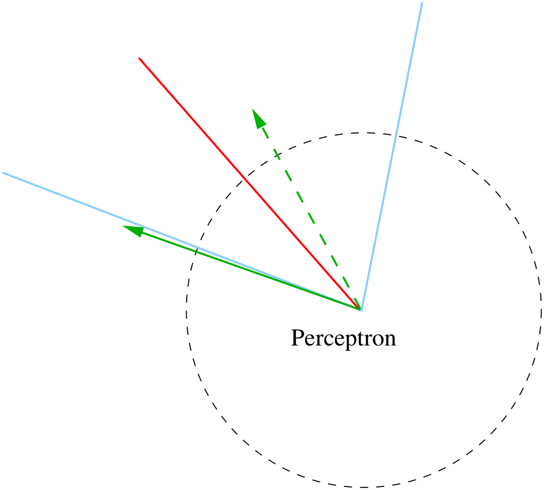
The cone of consistent hyperplanes



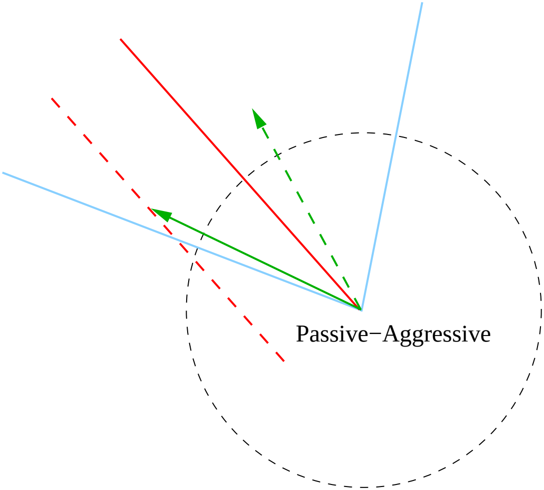
The cone of consistent hyperplanes



The cone of consistent hyperplanes



The cone of consistent hyperplanes



Mistake bounds for various updates

On any sequence of examples such that $y_t \mathbf{u}^\top \mathbf{x}_t \geqslant 1$ with $\|\mathbf{u}\| = U$

bound	algorithm	update time
U^2	Perceptron	$O(d)$

Mistake bounds for various updates

On any sequence of examples such that $y_t \mathbf{u}^\top \mathbf{x}_t \geqslant 1$ with $\|\mathbf{u}\| = U$

bound	algorithm	update time
U^2	Perceptron	$O(d)$
U^2	Passive-Aggressive	$O(d)$

Mistake bounds for various updates

On any sequence of examples such that $y_t \mathbf{u}^\top \mathbf{x}_t \geqslant 1$ with $\|\mathbf{u}\| = U$

bound	algorithm	update time
U^2	Perceptron	$O(d)$
U^2	Passive-Aggressive	$O(d)$
$dU \ln U$	2nd order Perceptron	$O(d^2)$

Mistake bounds for various updates

On any sequence of examples such that $y_t \mathbf{u}^\top \mathbf{x}_t \geqslant 1$ with $\|\mathbf{u}\| = U$

bound	algorithm	update time
U^2	Perceptron	$O(d)$
U^2	Passive-Aggressive	$O(d)$
$dU \ln U$	2nd order Perceptron	$O(d^2)$
$d^2 \ln U$	Ellipsoid	$O(d^3)$

Mistake bounds for various updates

On any sequence of examples such that $y_t \mathbf{u}^\top \mathbf{x}_t \geqslant 1$ with $\|\mathbf{u}\| = U$

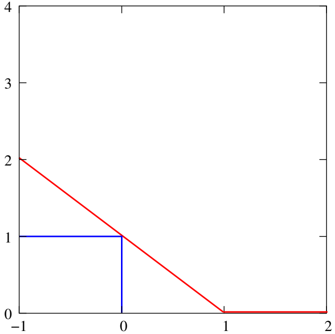
bound	algorithm	update time
U^2	Perceptron	$O(d)$
U^2	Passive-Aggressive	$O(d)$
$dU \ln U$	2nd order Perceptron	$O(d^2)$
$d^2 \ln U$	Ellipsoid	$O(d^3)$
$d \ln U$	Volumetric center	$O(d^{3.5})$

Mistake bounds for various updates

On any sequence of examples such that $y_t \mathbf{u}^\top \mathbf{x}_t \geqslant 1$ with $\|\mathbf{u}\| = U$

bound	algorithm	update time
U^2	Perceptron	$O(d)$
U^2	Passive-Aggressive	$O(d)$
$dU \ln U$	2nd order Perceptron	$O(d^2)$
$d^2 \ln U$	Ellipsoid	$O(d^3)$
$d \ln U$	Volumetric center	$O(d^{3.5})$
$d \ln U$	Geometric center	$O(d^4)$

The nonseparable case



$$\underbrace{\mathbb{I}_{\{\mathrm{SGN}(z)\neq y\}}}_{\text{mistake ind.}} \leqslant \underbrace{\left(1 - y\,z\right)_+}_{\text{hinge loss}}$$

- Computing an hyperplane minimizing the number of misclassified examples is NP-hard
- The hinge loss is a convex upper bound of the mistake indicator function

Perceptron mistake bound

Perceptron's performance is compared to the hinge loss of the single best linear classifier $\mathbf{u} \in \mathbb{R}^d$ in hindsight

Perceptron mistake bound

Perceptron's performance is compared to the hinge loss of the single best linear classifier $\mathbf{u} \in \mathbb{R}^d$ in hindsight

For any $\mathbf{u} \in \mathbb{R}^d$ and any sequence $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ define

total hinge loss $$D_{\mathbf{u}} = \sum_t \left(1 - y_t\, \mathbf{u}^\top \mathbf{x}_t\right)_+$$

Perceptron mistake bound

Perceptron's performance is compared to the hinge loss of the single best linear classifier $\mathbf{u} \in \mathbb{R}^d$ in hindsight

For any $\mathbf{u} \in \mathbb{R}^d$ and any sequence $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)$ define

total hinge loss $\qquad D_{\mathbf{u}} = \sum_t \left(1 - y_t\, \mathbf{u}^\top \boldsymbol{x}_t\right)_+$

On any sequence of examples, the number of mistakes made by the Perceptron is at most

$$\inf_{\mathbf{u} \in \mathbb{R}^d} \left(D_{\mathbf{u}} + \|\mathbf{u}\|^2 + \|\mathbf{u}\| \sqrt{D_{\mathbf{u}}} \right)$$

Perceptron mistake bound

Perceptron's performance is compared to the hinge loss of the single best linear classifier $\mathbf{u} \in \mathbb{R}^d$ in hindsight

For any $\mathbf{u} \in \mathbb{R}^d$ and any sequence $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ define

total hinge loss $$D_{\mathbf{u}} = \sum_t \left(1 - y_t\, \mathbf{u}^\top \mathbf{x}_t\right)_+$$

On any sequence of examples, the number of mistakes made by the Perceptron is at most

$$\inf_{\mathbf{u} \in \mathbb{R}^d} \left(D_{\mathbf{u}} + \|\mathbf{u}\|^2 + \|\mathbf{u}\| \sqrt{D_{\mathbf{u}}} \right)$$

Similar to the SVM functional $$\inf_{\mathbf{u} \in \mathbb{R}^d} \left(D_{\mathbf{u}} + \|\mathbf{u}\|^2 \right)$$

Summary

On-line learning with kernels

- Feature map $\phi : \mathbb{R}^d \to \mathrm{RKHS}$

On-line learning with kernels

- Feature map $\phi : \mathbb{R}^d \to \text{RKHS}$
- Kernel $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$

On-line learning with kernels

- Feature map $\phi : \mathbb{R}^d \to \mathrm{RKHS}$
- Kernel $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$
- Assume a linear algorithm learns $\boldsymbol{w}$ such that

$$\boldsymbol{w} = \sum_i \alpha_i \boldsymbol{x}_{t_i}$$

On-line learning with kernels

- Feature map $\phi : \mathbb{R}^d \to \text{RKHS}$
- Kernel $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$
- Assume a linear algorithm learns $\boldsymbol{w}$ such that

$$\boldsymbol{w} = \sum_i \alpha_i \boldsymbol{x}_{t_i}$$

- Then we can learn $w = \sum_i \alpha_i \phi(\boldsymbol{x}_{t_i})$ in the RKHS because

$$\begin{aligned}
\textsc{sgn}\big(\langle w, \phi(\boldsymbol{x})\rangle\big) &= \textsc{sgn}\left(\sum_i y_{t_i} \left\langle \phi(\boldsymbol{x}_{t_i}), \phi(\boldsymbol{x}) \right\rangle\right) \\
&= \textsc{sgn}\left(\sum_i y_{t_i} K(\boldsymbol{x}_{t_i}, \boldsymbol{x})\right)
\end{aligned}$$

Checking applicability of kernels

Let $\mathbf{R}_t = \sum_t y_t \mathbf{x}_t \mathbb{I}_{\{\widehat{p}_t \neq y_t\}}$

- Winnow $\quad w_{i,t} = \dfrac{e^{\eta\, R_{i,t}}}{\displaystyle\sum_{k=1}^{d} e^{\eta\, R_{k,t}}}$

Checking applicability of kernels

Let $\mathbf{R}_t = \sum_t y_t \mathbf{x}_t \mathbb{I}_{\{\widehat{p}_t \neq y_t\}}$

- Winnow $\quad w_{i,t} = \dfrac{e^{\eta\, R_{i,t}}}{\sum_{k=1}^{d} e^{\eta\, R_{k,t}}}$
- p-norm Perceptron $\quad w_{i,t} = \dfrac{\text{SGN}(R_{i,t})\, |R_{i,t}|^{p-1}}{\|\mathbf{R}_t\|_p^{p-2}}$

Checking applicability of kernels

Let $\mathbf{R}_t = \sum_t y_t \boldsymbol{x}_t \mathbb{I}_{\{\widehat{p}_t \neq y_t\}}$

- Winnow $\quad w_{i,t} = \dfrac{e^{\eta\, R_{i,t}}}{\displaystyle\sum_{k=1}^{d} e^{\eta\, R_{k,t}}}$
- p-norm Perceptron $\quad w_{i,t} = \dfrac{\mathrm{SGN}(R_{i,t})\,|R_{i,t}|^{p-1}}{\|\mathbf{R}_t\|_p^{p-2}}$
- Perceptron $\quad \boldsymbol{w}_t = \mathbf{R}_t$

Checking applicability of kernels

Let $\mathbf{R}_t = \sum_t y_t \boldsymbol{x}_t \mathbb{I}_{\{\widehat{p}_t \neq y_t\}}$

- Winnow $\quad w_{i,t} = \dfrac{e^{\eta\, R_{i,t}}}{\displaystyle\sum_{k=1}^{d} e^{\eta\, R_{k,t}}}$
- p-norm Perceptron $\quad w_{i,t} = \dfrac{\text{SGN}(R_{i,t})\,|R_{i,t}|^{p-1}}{\|\mathbf{R}_t\|_p^{p-2}}$
- Perceptron $\quad \boldsymbol{w}_t = \mathbf{R}_t$

Perceptron's potential is spherical $\longrightarrow$ rotational invariance

Kernel Perceptron

Start with empty cache $\mathcal{L}$ of examples

Loop:

Kernel Perceptron

Start with empty cache $\mathcal{L}$ of examples

Loop:

1. Read next instance x_t

Kernel Perceptron

Start with empty cache $\mathcal{L}$ of examples

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \text{SGN}\left(\sum_{\boldsymbol{v}\in\mathcal{L}} K(\boldsymbol{v}, \boldsymbol{x}_t)\right)$

Kernel Perceptron

Start with empty cache $\mathcal{L}$ of examples

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \text{SGN}\left(\sum_{\boldsymbol{v}\in\mathcal{L}} K(\boldsymbol{v}, \boldsymbol{x}_t)\right)$
3. Obtain true label y_t

Kernel Perceptron

Start with empty cache $\mathcal{L}$ of examples

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \operatorname{SGN}\left(\sum_{\boldsymbol{v} \in \mathcal{L}} K(\boldsymbol{v}, \boldsymbol{x}_t)\right)$
3. Obtain true label y_t
4. If $\widehat{p}_t \neq y_t$ (mistake) then store new support $(y_t \boldsymbol{x}_t)$ in $\mathcal{L}$

Kernel Perceptron

Start with empty cache $\mathcal{L}$ of examples

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \operatorname{SGN}\left(\sum_{\boldsymbol{v}\in\mathcal{L}} K(\boldsymbol{v}, \boldsymbol{x}_t)\right)$
3. Obtain true label y_t
4. If $\widehat{p}_t \neq y_t$ (mistake) then store new support $(y_t \boldsymbol{x}_t)$ in $\mathcal{L}$

Mistake bounds hold in the whole RKHS

Memory bounded learning

Can we control the rate of mistakes when at most $B < \infty$ supports are used?

Memory bounded learning

Can we control the rate of mistakes when at most $B < \infty$ supports are used?

Fact

Using at most B supports, any learner makes an unbounded number of mistakes on a sequence that is perfectly classified by some $\mathbf{u} \in \mathbb{R}^d$ with zero hinge loss and $\|\mathbf{u}\| = \sqrt{B+1}$

Memory bounded learning

Can we control the rate of mistakes when at most $B < \infty$ supports are used?

Fact

Using at most B supports, any learner makes an unbounded number of mistakes on a sequence that is perfectly classified by some $\mathbf{u} \in \mathbb{R}^d$ with zero hinge loss and $\|\mathbf{u}\| = \sqrt{B+1}$

- Thus $B \geqslant U^2$ is necessary to compete against $\mathbf{u}$ of length U

Memory bounded learning

Can we control the rate of mistakes when at most $B < \infty$ supports are used?

Fact

Using at most B supports, any learner makes an unbounded number of mistakes on a sequence that is perfectly classified by some $\mathbf{u} \in \mathbb{R}^d$ with zero hinge loss and $\|\mathbf{u}\| = \sqrt{B+1}$

- Thus $B \geqslant U^2$ is necessary to compete against $\mathbf{u}$ of length U
- Can we compete against any $\mathbf{u}$ with $\|\mathbf{u}\| \leqslant U$ using $B = (1+\varepsilon)U^2$ supports?

A randomized perceptron

Randomized Budget Perceptron

Parameter: size B of cache for supports

Start with empty cache $\mathcal{L}$

A randomized perceptron

Randomized Budget Perceptron

Parameter: size B of cache for supports

Start with empty cache $\mathcal{L}$

Loop:

1. Read next instance $\boldsymbol{x}_t$

A randomized perceptron

Randomized Budget Perceptron

Parameter: size B of cache for supports

Start with empty cache $\mathcal{L}$

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \text{SGN}\left(\sum_{\boldsymbol{v} \in \mathcal{L}} \boldsymbol{v}^\top \boldsymbol{x}_t\right)$

A randomized perceptron

Randomized Budget Perceptron

Parameter: size B of cache for supports

Start with empty cache $\mathcal{L}$

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \text{SGN}\left(\sum_{\boldsymbol{v}\in\mathcal{L}} \boldsymbol{v}^\top \boldsymbol{x}_t\right)$
3. Obtain true label y_t

A randomized perceptron

Randomized Budget Perceptron

Parameter: size B of cache for supports

Start with empty cache $\mathcal{L}$

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \mathrm{SGN}\left(\sum_{\boldsymbol{v}\in\mathcal{L}} \boldsymbol{v}^\top \boldsymbol{x}_t\right)$
3. Obtain true label y_t
4. If $\widehat{p}_t \neq y_t$ then:

A randomized perceptron

Randomized Budget Perceptron

Parameter: size B of cache for supports

Start with empty cache $\mathcal{L}$

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \text{SGN}\left(\sum_{\boldsymbol{v}\in\mathcal{L}} \boldsymbol{v}^\top \boldsymbol{x}_t\right)$
3. Obtain true label y_t
4. If $\widehat{p}_t \neq y_t$ then:
   1. If $|\mathcal{L}| = B$, then throw away a random support from $\mathcal{L}$

A randomized perceptron

Randomized Budget Perceptron

Parameter: size B of cache for supports

Start with empty cache $\mathcal{L}$

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \text{SGN}\left(\sum_{\boldsymbol{v}\in\mathcal{L}} \boldsymbol{v}^\top \boldsymbol{x}_t\right)$
3. Obtain true label y_t
4. If $\widehat{p}_t \neq y_t$ then:
    1. If $|\mathcal{L}| = B$, then throw away a random support from $\mathcal{L}$
    2. Add $y_t \boldsymbol{x}_t$ to $\mathcal{L}$

A randomized perceptron

Randomized Budget Perceptron

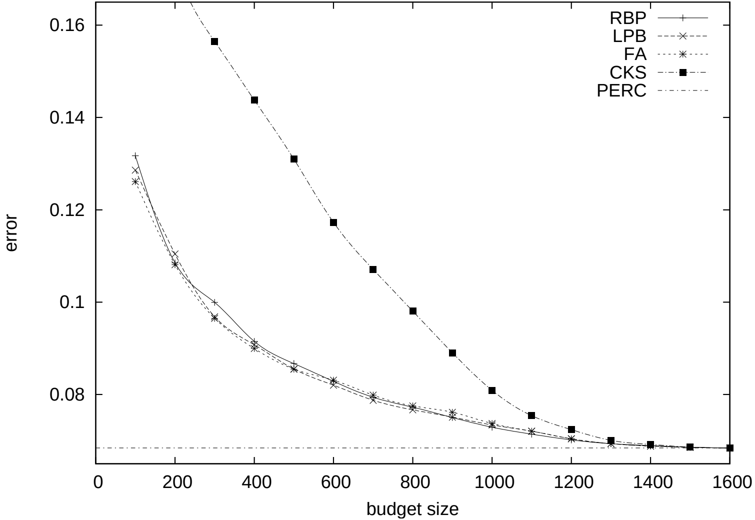
Parameter: size B of cache for supports
Start with empty cache $\mathcal{L}$
Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \text{SGN}\left(\sum_{\boldsymbol{v}\in\mathcal{L}} \boldsymbol{v}^\top \boldsymbol{x}_t\right)$
3. Obtain true label y_t
4. If $\widehat{p}_t \neq y_t$ then:
    1. If $|\mathcal{L}| = B$, then throw away a random support from $\mathcal{L}$
    2. Add $y_t \boldsymbol{x}_t$ to $\mathcal{L}$

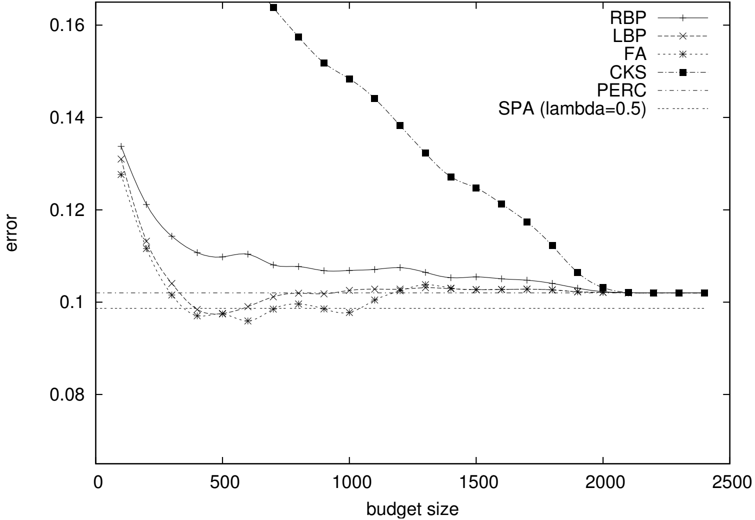
Result:

Bound on mistakes scales roughly with $1 + 1/\varepsilon$

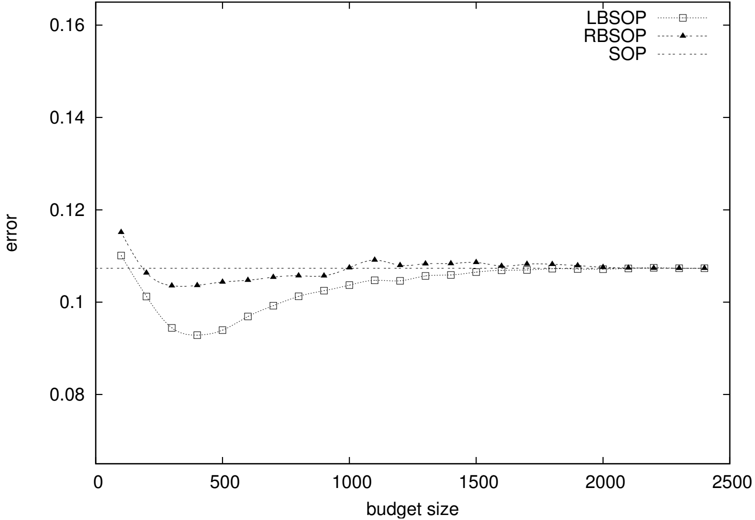
Empirical performance — stationary



Empirical performance — nonstationary



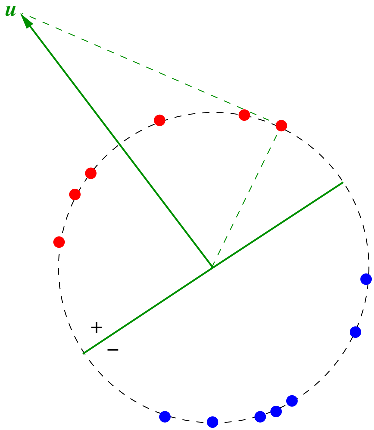
Empirical performance 2nd order — nonstationary



Summary

Online approximation of SVM hyperplane

The SVM hyperplane is the shortest $\mathbf{u}$ such that $y_t \mathbf{u}^\top \mathbf{x}_t \geqslant 1$ for all t



Online approximation of SVM hyperplane (cont.)

The ALMA algorithm

Parameter: $0 < \alpha \leqslant 1$

Set mistake counter $k = 1$

Online approximation of SVM hyperplane (cont.)

The ALMA algorithm

Parameter: $0 < \alpha \leqslant 1$

Set mistake counter $k = 1$

Loop:

1. Read next instance $\boldsymbol{x}_t$

Online approximation of SVM hyperplane (cont.)

The ALMA algorithm

Parameter: $0 < \alpha \leqslant 1$

Set mistake counter $k = 1$

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \textsc{sgn}\big(\boldsymbol{w}^\top \boldsymbol{x}_t\big)$

Online approximation of SVM hyperplane (cont.)

The ALMA algorithm

Parameter: $0 < \alpha \leqslant 1$

Set mistake counter $k = 1$

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \textsc{sgn}\left(\boldsymbol{w}^\top \boldsymbol{x}_t\right)$
3. Obtain true label y_t

Online approximation of SVM hyperplane (cont.)

The ALMA algorithm

Parameter: $0 < \alpha \leqslant 1$

Set mistake counter $k = 1$

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \operatorname{SGN}\big(\boldsymbol{w}^\top \boldsymbol{x}_t\big)$
3. Obtain true label y_t
4. If margin smaller than $c(1-\alpha)/\sqrt{k}$ then:

Online approximation of SVM hyperplane (cont.)

The ALMA algorithm

Parameter: $0 < \alpha \leqslant 1$

Set mistake counter $k = 1$

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \text{SGN}\big(\boldsymbol{w}^\top \boldsymbol{x}_t\big)$
3. Obtain true label y_t
4. If margin smaller than $c(1-\alpha)/\sqrt{k}$ then:
    1. $\boldsymbol{w}' = \boldsymbol{w} + y_t \boldsymbol{x}_t/\sqrt{k}$

Online approximation of SVM hyperplane (cont.)

The ALMA algorithm

Parameter: $0 < \alpha \leqslant 1$

Set mistake counter $k = 1$

Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \text{SGN}\big(\boldsymbol{w}^\top \boldsymbol{x}_t\big)$
3. Obtain true label y_t
4. If margin smaller than $c(1-\alpha)\big/\sqrt{k}$ then:
   1. $\boldsymbol{w}' = \boldsymbol{w} + y_t \boldsymbol{x}_t \big/ \sqrt{k}$
   2. $\boldsymbol{w} = \boldsymbol{w}' \big/ \|\boldsymbol{w}'\| \qquad k \leftarrow k+1$

Online approximation of SVM hyperplane (cont.)

The ALMA algorithm

Parameter: $0 < \alpha \leqslant 1$

Set mistake counter $k = 1$

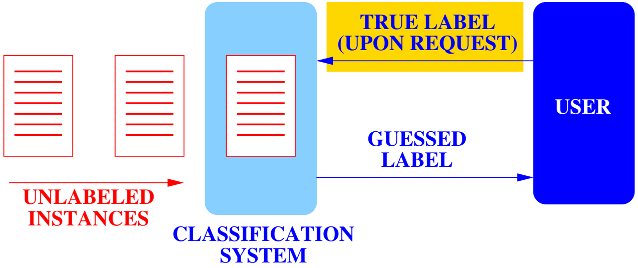
Loop:

1. Read next instance $\boldsymbol{x}_t$
2. Predict y_t with $\widehat{p}_t = \textsc{sgn}\big(\boldsymbol{w}^\top \boldsymbol{x}_t\big)$
3. Obtain true label y_t
4. If margin smaller than $c(1-\alpha)\big/\sqrt{k}$ then:
    1. $\boldsymbol{w}' = \boldsymbol{w} + y_t \boldsymbol{x}_t \big/ \sqrt{k}$
    2. $\boldsymbol{w} = \boldsymbol{w}' \big/ \left\| \boldsymbol{w}' \right\| \qquad k \leftarrow k+1$

Result

Finds separating $\boldsymbol{u}$ with $\|\boldsymbol{u}\| \leqslant \|\boldsymbol{u}_{\text{SVM}}\| \big/ (1-\alpha)$ after at most $\big(\|\boldsymbol{u}_{\text{SVM}}\| \big/ \alpha\big)^2$ updates

Selective sampling



A selective sampling classifier

1. Classify next instance $\boldsymbol{x}_t$ with $\operatorname{SGN}(\boldsymbol{w}^\top \boldsymbol{x}_t)$

A selective sampling classifier

1. Classify next instance $\boldsymbol{x}_t$ with $\operatorname{SGN}(\boldsymbol{w}^\top \boldsymbol{x}_t)$
2. If $|\boldsymbol{w}^\top \boldsymbol{x}_t| \leqslant \|\boldsymbol{x}_t\| \sqrt{\frac{c \ln t}{N_t}}$ then query label y_t of $\boldsymbol{x}_t$

N_t = number of labels sampled so far

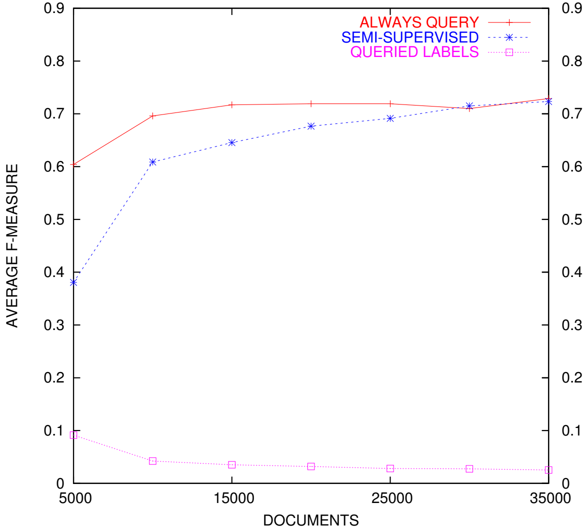
A selective sampling classifier

1. Classify next instance $\boldsymbol{x}_t$ with $\text{SGN}(\boldsymbol{w}^\top \boldsymbol{x}_t)$
2. If $|\boldsymbol{w}^\top \boldsymbol{x}_t| \leqslant \|\boldsymbol{x}_t\| \sqrt{\dfrac{c \ln t}{N_t}}$ then query label y_t of $\boldsymbol{x}_t$
3. If label queried then use $(\boldsymbol{x}_t, y_t)$ to update $\boldsymbol{w}$

N_t = number of labels sampled so far

$\boldsymbol{w}$ updated with the 2nd order Perceptron update rule

Empirical performance on RCV1



Summary

1. Prediction with expert advice
2. Linear classification
3. Kernel-based on-line learning
4. Online SVM and active learning
5. From mistake to risk bounds

Statistical learning theory

- Linear classifiers $H(\boldsymbol{x}) = \mathrm{SGN}\big(\boldsymbol{w}^\top \boldsymbol{x}\big)$

Statistical learning theory

- Linear classifiers $H(\boldsymbol{x}) = \mathrm{SGN}\big(\boldsymbol{w}^\top \boldsymbol{x}\big)$
- Examples $(\boldsymbol{x}_t, y_t)$ are i.i.d. according to a fixed and unknown probability distribution on $\mathbb{R}^d \times \{-1, +1\}$

Statistical learning theory

- Linear classifiers $H(\boldsymbol{x}) = \textsc{sgn}\big(\boldsymbol{w}^\top \boldsymbol{x}\big)$
- Examples $(\boldsymbol{x}_t, y_t)$ are i.i.d. according to a fixed and unknown probability distribution on $\mathbb{R}^d \times \{-1, +1\}$
- $\texttt{risk}(H) = \mathbb{P}\big(H(\boldsymbol{x}) \neq y\big)$

Statistical learning theory

- Linear classifiers $H(x) = \text{SGN}(\boldsymbol{w}^\top \boldsymbol{x})$
- Examples $(\boldsymbol{x}_t, y_t)$ are i.i.d. according to a fixed and unknown probability distribution on $\mathbb{R}^d \times \{-1, +1\}$
- $\mathtt{risk}(H) = \mathbb{P}\big(H(\boldsymbol{x}) \neq y\big)$
- Learning algorithm

$$(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n) \longrightarrow \boxed{A} \longrightarrow \widehat{H} : \mathbb{R}^d \to \{-1, +1\}$$

$\widehat{H}$ is (random) hypothesis output by learner

The ensemble of hypotheses

- Run an incremental learner on the training set

The ensemble of hypotheses

- Run an incremental learner on the training set
- Everytime $H(x_t) \neq y_t$, H is changed by the update rule

The ensemble of hypotheses

- Run an incremental learner on the training set
- Everytime $H(\boldsymbol{x}_t) \neq y_t$, H is changed by the update rule
- This process generates an ensemble of classifiers

$$H_0, H_1, \ldots, H_n$$

The ensemble of hypotheses

- Run an incremental learner on the training set
- Everytime $H(\mathbf{x}_t) \neq y_t$, H is changed by the update rule
- This process generates an ensemble of classifiers

$$H_0, H_1, \dots, H_n$$

Goals

1. Bound the average risk of the ensemble in terms of the size of the ensemble

The ensemble of hypotheses

- Run an incremental learner on the training set
- Everytime $H(\boldsymbol{x}_t) \neq y_t$, H is changed by the update rule
- This process generates an ensemble of classifiers

$$H_0, H_1, \dots, H_n$$

Goals

1. Bound the average risk of the ensemble in terms of the size of the ensemble
2. Find an element of the ensemble whose risk is close to the ensemble average

Step 1: bound the average risk

The difference

$$\mathtt{risk}(H_{t-1}) - \mathbb{I}_{\{H_{t-1}(x_t) \neq y_t\}}$$

is a martingale difference sequence because

$$\mathbb{E}\Big[\mathtt{risk}(H_{t-1}) - \mathbb{I}_{\{H_{t-1}(x_t) \neq y_t\}} \,\Big|\, (x_1, y_1), \dots, (x_{t-1}, y_{t-1})\Big] = 0$$

Step 1: bound the average risk

The difference

$$\mathrm{risk}(H_{t-1}) - \mathbb{I}_{\{H_{t-1}(x_t) \neq y_t\}}$$

is a martingale difference sequence because

$$\mathbb{E}\left[\mathrm{risk}(H_{t-1}) - \mathbb{I}_{\{H_{t-1}(x_t) \neq y_t\}} \,\middle|\, (x_1, y_1), \dots, (x_{t-1}, y_{t-1})\right] = 0$$

The associated martingale is

$$\sum_{t=1}^{n} \left(\mathrm{risk}(H_{t-1}) - \mathbb{I}_{\{H_{t-1}(x_t) \neq y_t\}}\right)$$

$$\Longleftrightarrow \quad \underbrace{\frac{1}{n} \sum_{t=1}^{n} \mathrm{risk}(H_{t-1})}_{\text{average risk}} - \underbrace{\frac{1}{n} \sum_{t=1}^{n} \mathbb{I}_{\{H_{t-1}(x_t) \neq y_t\}}}_{\text{fraction of mistakes}}$$

Bernstein's bound

If $Z_1, Z_2, \dots$ is a martingale difference sequence with increments bounded by 1 and

$$V_n = \sum_{t=1}^{n} \mathbb{E}\left[Z_t^2 \mid Z_1, \dots, Z_{t-1}\right]$$

then for all $S, K > 0$

$$\mathbb{P}\left(\sum_{t=1}^{n} Z_n \geqslant S, \quad V_n \leqslant K\right) \leqslant \exp\left(-\frac{S^2}{2(S/3 + K)}\right)$$

Application of Bernstein's bound

Since $0 \leqslant \mathbb{I}_{\{H(x) \neq y\}} \leqslant 1$,

$$
\begin{aligned}
& \mathrm{VAR}\left[\mathbb{I}_{\{H_{t-1}(x_t),y_t\}} \,\middle|\, (x_1,y_1),\dots,(x_{t-1},y_{t-1})\right] \\
\leqslant \quad & \mathbb{E}\left[\mathtt{risk}(H_{t-1}) \,\middle|\, (x_1,y_1),\dots,(x_{t-1},y_{t-1})\right]
\end{aligned}
$$

Application of Bernstein's bound

Since $0 \leqslant \mathbb{I}_{\{H(x) \neq y\}} \leqslant 1$,

$$\begin{aligned} &\text{VAR}\Big[\mathbb{I}_{\{H_{t-1}(x_t), y_t\}} \,\Big|\, (x_1, y_1), \dots, (x_{t-1}, y_{t-1})\Big] \\ \leqslant \quad &\mathbb{E}\Big[\texttt{risk}(H_{t-1}) \,\Big|\, (x_1, y_1), \dots, (x_{t-1}, y_{t-1})\Big] \end{aligned}$$

Applying Bernstein's gives

$$\frac{1}{n}\sum_{t=1}^{n} \texttt{risk}(H_{t-1}) \quad \leqslant \quad \frac{M_n}{n} + \frac{c}{n}\left(\ln M_n + \sqrt{M_n \ln M_n}\right) \quad \text{w.h.p.}$$

Where $\displaystyle \frac{M_n}{n} = \frac{1}{n}\sum_{t=1}^{n} \mathbb{I}_{\{H_{t-1}(x_t) \neq Y_t\}}$ is the fraction of mistakes

Step 2: pick a good classifier in the ensemble

- Start from the ensemble $H_0, H_1, \dots, H_n$
- Do the following:
    1. test each H_t on $(\boldsymbol{x}_{t+1}, y_{t+1}), \dots, (\boldsymbol{x}_n, y_n)$
    2. pick $\widehat{H} = H_{t^*}$ minimizing a penalized risk estimate

Step 2: pick a good classifier in the ensemble

- Start from the ensemble $H_0, H_1, \ldots, H_n$
- Do the following:
  1. test each H_t on $(\boldsymbol{x}_{t+1}, y_{t+1}), \ldots, (\boldsymbol{x}_n, y_n)$
  2. pick $\widehat{H} = H_{t^*}$ minimizing a penalized risk estimate

Guaranteed bound

$$\mathtt{risk}(\widehat{H}) \quad \leqslant \quad \frac{M_n}{n} + \frac{c}{n}\left((\ln n)^2 + \sqrt{M_n \ln n}\right) \qquad \text{w.h.p.}$$

Conclusions

- A game-theoretic foundation for on-line learning

Conclusions

- A game-theoretic foundation for on-line learning
- Performance guarantees for several variants of the basic model

Conclusions

- A game-theoretic foundation for on-line learning
- Performance guarantees for several variants of the basic model
- Learning with structured outputs builds naturally on these results