
Combining expert advice

Jyrki Kivinen

Seminar talk 21 November 2006

Basic setting

We want to make predictions about some sequence y_1, \dots, y_T of **outcomes** (binary classes, real values, actions, ...)

To help us, we have **experts** $\mathcal{E}_1, \dots, \mathcal{E}_N$.

Examples:

- Fixed parametric model $P(Y|\theta)$ and N different parameter settings $\theta_1, \dots, \theta_N$. Expert \mathcal{E}_i believes that y_t follows distribution $P(Y|\theta_i)$.
- N different learning algorithms. Expert \mathcal{E}_i is using algorithm number i .
- Single learning algorithm with “tuning parameter” η . Expert \mathcal{E}_i is using $\eta = \eta_i$, for some finite range of values η_1, \dots, η_N .
- N stocks in the market. Expert \mathcal{E}_i says we should always put our money in stock number i .

Questions:

- How to combine the advice into a “good” overall prediction?
- What is “good”?

We try to (learn to) combine expert predictions:

- consider each expert as black box that produces predictions
- **ignore** experts' internal working, and also their **input**
- We do **not** consider settings where we know the inputs of the experts and could use that “side information” to favour different experts for different inputs.

Need to be careful defining performance criteria:

- Sometimes no combination will predict well:
 - all experts are bad or
 - problem is inherently noisy.
- If all experts are good, almost any combination will predict well.
- Interesting case: many experts, one (or few) much better than others; find the good ones

Notation

$N \in \mathbf{N}$: number of experts

$i, j \in \{1, \dots, N\}$: indices for experts

$T \in \mathbf{N}$: time horizon (“number of data points”)

$t, \tau \in \{1, \dots, T\}$: indices for time steps

X : range of expert predictions

Y : range of outcomes

Here we usually take $X = Y$. Typical examples:

binary prediction $X = Y = \{0, 1\}$; regression $X = Y = [a, b] \subset \mathbf{R}$ for some $a, b \in \mathbf{R}$.

$y_t \in Y$: outcome at time t

$x_{t,i} \in X$: prediction of expert i about outcome y_t

$\mathbf{x}_t \in X^N$: vector of expert predictions at time t

$\hat{y}_t \in X$: our combined prediction about y_t

$L: Y \times X \rightarrow [0, \infty)$: loss function

Typical examples: $L(y, \hat{y}) = (y - \hat{y})^2$ (for regression);

$L(y, \hat{y}) = 0$ if $y = \hat{y}$ and $L(y, \hat{y}) = 1$ otherwise (for classification)

On-line prediction

Consider a protocol where the **Learner** (\approx our algorithm) tries to combine expert advice. Learner's input is provided by the **Environment**.

The following is repeated for $t = 1, \dots, T$:

1. **Environment** chooses and reveals the experts' predictions $\mathbf{x}_t \in X^N$.
2. **Environment** chooses, but does not reveal, the outcome $y_t \in Y$.
3. **Learner** chooses and reveals its prediction $\hat{y}_t \in X$.
4. **Environment** reveals the outcome $y_t \in Y$.

Learner's prediction \hat{y}_t may depend on \mathbf{x}_τ and y_τ for $\tau < t$, and on \mathbf{x}_t .

Environment can choose arbitrary \mathbf{x}_t and y_t (but they can't depend on \hat{y}_τ for $\tau \geq t$).

After T timesteps, we calculate **total loss** for Learner and each expert:

$$\begin{aligned} \text{Loss}(\text{Learner}) &= \sum_{t=1}^T L(y_t, \hat{y}_t) \\ \text{Loss}(\mathcal{E}_i) &= \sum_{t=1}^T L(y_t, x_{t,i}) \quad \text{for } i = 1, \dots, N. \end{aligned}$$

The **regret** of the learner is the excess loss over the best single expert:

$$\text{Loss}(\text{Learner}) - \min_{1 \leq i \leq N} \text{Loss}(\mathcal{E}_i).$$

Typical results we can obtain are of the form

$$\text{Loss}(\text{Learner}) - \min_{1 \leq i \leq N} \text{Loss}(\mathcal{E}_i) \leq c \log N \quad \text{or}$$

$$\text{Loss}(\text{Learner}) - \min_{1 \leq i \leq N} \text{Loss}(\mathcal{E}_i) \leq b\sqrt{T \log N} + c \log N$$

where b and c are (small) constants that depend only on the loss function. This means that the **regret per timestep** converges to zero:

$$\frac{1}{T} \left(\text{Loss}(\text{Learner}) - \min_{1 \leq i \leq N} \text{Loss}(\mathcal{E}_i) \right) = O \left(\frac{\log N}{T} \right) \quad \text{or}$$

$$\frac{1}{T} \left(\text{Loss}(\text{Learner}) - \min_{1 \leq i \leq N} \text{Loss}(\mathcal{E}_i) \right) = O \left(\sqrt{\frac{\log N}{T}} \right).$$

Important: Bounds hold for all choices of Environment; no statistical assumptions involved!

Aggregating algorithm [Vovk, 1989; Littlestone & Warmuth, 1989]

This algorithm for the Learner maintains a **weight** for each expert. The weight of expert \mathcal{E}_i at timestep t is denoted by $w_{t,i}$.

The aggregating algorithm as Learner works as follows:

- Initially $w_{1,i} = 1$ for $i = 1, \dots, N$.
- At time t , choose a **random index** $i \in \{1, \dots, N\}$ so that the probability of choosing i is proportional to $w_{t,i}$. Make prediction $\hat{y}_t = x_{t,i}$.
- After y_t is observed, **update weights** according to

$$w_{t+1,i} = w_{t,i} \exp(-\eta L(y_t, x_{t,i})),$$

where parameter $\eta > 0$ is the **learning rate**.

Thus

$$-\ln w_{t,i} = \eta \sum_{\tau=1}^{t-1} L(y_\tau, x_{\tau,i})$$

which can give a probabilistic interpretation to the weights.

Assuming bounded loss function and suitable η , the aggregating algorithm achieves

$$E [\text{Loss}(\text{Learner})] - \min_{1 \leq i \leq N} \text{Loss}(\mathcal{E}_i) \leq b\sqrt{T \log N} + c \log N$$

where the expectation is over random choices of the algorithm.

- Bounds that hold with high probability can also be obtained.
- For regression we can get

$$\text{Loss}(\text{Learner}) - \min_{1 \leq i \leq N} \text{Loss}(\mathcal{E}_i) \leq c \log N$$

with a **deterministic** algorithm that predicts

$$\hat{y}_t = \sum_{i=1}^N \left(\frac{w_{t,i}}{\sum_{j=1}^N w_{t,j}} \right) \cdot x_{t,i}.$$

Follow the perturbed leader [Hannan 1957]

An alternative algorithm: At time t ,

1. Draw N independent random variables $Z_{t,i} \in \mathbf{R}$, $i = 1, \dots, N$, from some distribution P .
2. Predict $\hat{y}_t = x_{t,i}$, where i minimises

$$Z_{t,i} + \sum_{\tau=1}^{t-1} L(y_{\tau}, x_{\tau,i}).$$

The basic “follow the leader” would be to choose i that minimises

$$\sum_{\tau=1}^{t-1} L(y_{\tau}, x_{\tau,i});$$

here we add a random perturbation $Z_{t,i}$ to avoid bad worst-case behaviour. With suitable P , regret bounds are similar to the Aggregating Algorithm (details omitted).

From prediction to reinforcement learning

In principle, expert \mathcal{E}_i could be a “policy”, and each $x_{t,i}$ an “action”. Then $-L(y_t, x_{t,i})$ is the “payoff” for that action. Problems:

1. If actions affect the state of the system, changing policies at every step does not make sense.
2. If we choose action $x_{t,i}$, we don't see the payoffs for the other actions.
3. In particular, if we always choose actions that “look good”, we may miss finding new, even better ones (exploration-exploitation dilemma).

For problem 1 there are currently only very rudimentary solutions [Peter Auer and others, ca. 2005]. We present some ideas for 2 and 3 (also due to Auer et al.) in context of the [multiarmed bandit problem](#).

Multiarmed bandit problem

We have a set $\{1, \dots, N\}$ of actions. Choosing action i at time t would give loss $L(t, i)$.

Consider the protocol where at time t

1. Environment chooses, but does not reveal, $L(t, i)$ for $i = 1, \dots, N$.
2. Learner chooses action $a_t \in \{1, \dots, N\}$.
3. Environment reveals $L(t, a_t)$.

As in prediction, Learner tries to minimise regret

$$\sum_{t=1}^T L(t, a_t) - \min_{1 \leq i \leq N} \sum_{t=1}^T L(t, i).$$

However, now $L(t, i)$ is known only if $i = a_t$.

Aggregating algorithm for bandits

[Auer, Cesa-Bianchi, Freund & Schapire 2002]

Like in prediction, maintain weights $w_{t,i}$.

- Initially $w_{1,i} = 1$ for $i = 1, \dots, N$.
- At time t , pick action $a_t \in \{1, \dots, N\}$ so that probability of choosing $a_t = i$ is

$$p_{t,i} = (1 - \gamma) \frac{w_{t,i}}{\sum_{j=1}^N w_{t,j}} + \frac{\gamma}{N}.$$

- After $L(t, a_t)$ is observed, update

$$w_{t+1,i} = w_{t,i} \exp(-\eta \tilde{L}(t, i)) \quad \text{for } i = 1, \dots, N,$$

where

$$\tilde{L}(t, i) = \begin{cases} L(t, i)/p_{t,i} & \text{if } i = a_t \\ 0 & \text{otherwise.} \end{cases}$$

Key property: $\tilde{L}(t, i)$ is an unbiased estimate of $L(t, i)$.

Parameter γ controls exploration-exploitation tradeoff.

Confidence bounds for bandits [Auer 2002]

We sketch the idea of confidence bound algorithms in a simple setting where for each i there is a fixed but unknown distribution P_i such that $L_{t,i} \sim P_i$ for all t .

Learner maintains for each i a **lower bound** q_i such that $E[L_{t,i}] \geq q_i$ holds with high probability. We want q_i to be as close to $E[L_{t,i}]$ as possible.

Learner chooses $a_t = i$ where i minimises q_i .

Thus action i is likely to be chosen, if

- we are confident that i is good or
- we are very uncertain whether i is good or poor. This balances exploration and exploitation.

Pointers to literature

Nicolò Cesa-Bianchi and Gábor Lugosi (2006). *Prediction, Learning, and Games*. Cambridge University Press.

Yoav Freund's lecture notes <http://seed.ucsd.edu/down/Onlinecrs/CoursePlan.html>

Peter Auer (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3:397–422.

Yoav Freund and Robert E. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55:119–139.

Nick Littlestone and Manfred K. Warmuth (1994). The weighted majority algorithm. *Information and Computation* 108:212–261.

Volodya Vovk (1998). A game of prediction with expert advice. *Journal of Computer and System Sciences* 56:153–173.