On-line Learning in Santa Cruz

Manfred K. Warmuth

University of California - Santa Cruz

CMPS 200, W15
What is our research group known for

- On-line learning
- Multiplicative updates
- Regret bounds
- Applications of on-line algorithms

Main conferences: COLT, ICML, NIPS, UAI
Goals for this talk

- Philosophical problems in Machine Learning
  - Occam’s razor
  - Over fitting
Goals for this talk

- Philosophical problems in Machine Learning
  - Occam’s razor
  - Over fitting

- Give a sense of the
  - problems
  - algorithms
  - analysis techniques
Linear Regression

Examples  \((x_t, y_t) \in \mathbb{R}^n \times \mathbb{R}\)

Curve fitting  \(\min_w \sum_t (w \cdot x_t - y_t)^2\)
Fitting data

\[ g(x) \]: degree 1  \[ h(x) \]: degree 2

degree 8: perfect fit?
First deep philosophical problem

Best fit not necessarily best
**Best fit not necessarily best**

- High degree curve give best fit, but it does not “learn” / “generalize”.
  Just stores the data

- **Overfitting**

- How do you prevent it?
Machine learning solution for batch learning

Split batch into training and test set

**Training error - Test error**

![Graph showing training and test errors vs complexity]

- **y-axis:** error  -  **x-axis:** *complexity*

Trade-off between simplicity (Occam’s razor) and fit
What notion of complexity

- Degree of polynomial
- Complex: large training time - simple: early stopping
- Regularization:

$$\min_w \left\| w \right\|_2^2 + \eta \sum_{t \in \text{training set}} (w \cdot x_t - y_t)^2$$

Tune with validation set
Two main update families - linear regression

- **On-line Additive**

  \[ w_{t+1} = w_t - \eta \left( w_t \cdot x_t - y_t \right)x_t \]

  **Gradient of Square Loss**

  - **Regularizer:** Squared Euclidean Distance \( \|w\|^2 \)
  - **Weights can go negative**
  - **Called Gradient Descent (GD)**

- **On-line Multiplicative**

  \[ w_{t+1,i} = \frac{w_{t,i} e^{-\eta(w_t \cdot x_t - y_t)x_t,i}}{Z_t} \]

  - **Regularizer:** relative entropy
  - **Updated weight vector stays on probability simplex**
  - **Called Exponentiated Gradient (EG)**
  - **Related to multiplicative updates of evolutionary processes**
Alternate to regularization

- Perturb the data with noise
- Prevents over fitting
  - The noise blurs out the details
- Recently proved good generalization for dropout
- What noise corresponds to what regularization?
Feed forward neural net
Weights parameters - sigmoids at internal nodes
Dropout training

- Stochastic gradient descent
- Randomly remove every hidden/input unit with prob. $\frac{1}{2}$ before each gradient descent update

[Hinton et al. 2012]
Dropout training

- Very successful in image recognition & speech recognition
- Why does it work?
  
  [Wagner, Wang, Liang 2013]
  [Helmbold, Long 2014]
What are we doing?

Prove bounds for dropout
- single neuron
- linear loss
On-line learning

$n$ experts perform a prediction task in each trial
Master algorithm combines experts with goal of performing as well as the best
On-line learning

$n$ experts perform a prediction task in each trial
Master algorithm combines experts with goal of performing as well as the best

Trial $t$
- predict with a distribution $w_t$ over the $n$ experts
- expert $i$ chosen with probability $w_{t,i}$
- get $n$ dimensional loss vector $\ell_t$
- expert $i$ has loss $\ell_{t,i}$
- algorithm’s loss is $\sum_i w_{t,i} \ell_{t,i} = w_t \cdot \ell_t$
- update $w_t \rightarrow w_{t+1}$
$n$ experts perform a prediction task in each trial
Master algorithm combines experts with goal of performing as well as the best

Trial $t$
- predict with a distribution $\mathbf{w}_t$ over the $n$ experts
- expert $i$ chosen with probability $w_{t,i}$
- get $n$ dimensional loss vector $\mathbf{l}_t$
- expert $i$ has loss $\ell_{t,i}$
- algorithm’s loss is $\sum_i w_{t,i} \ell_{t,i} = \mathbf{w}_t \cdot \mathbf{\ell}_t$
- update $\mathbf{w}_t \rightarrow \mathbf{w}_{t+1}$
How do we measure performance

Worst-case regret after $T$ trials:

$$
\sum_{t=1}^{T} w_t \cdot \ell_t - \inf_i \ell_{\leq T,i}^{\leq T,i}
$$

- total expected loss of alg
- loss $\ell^*$ of best expert

Should be logarithmic in $\#$ of experts $n$
### Main algorithms

<table>
<thead>
<tr>
<th></th>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$E_3$</th>
<th>$E_4$</th>
<th>$E_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**day $t - 1$**

$\ell_{\leq t-1,i}$
### Main algorithms

<table>
<thead>
<tr>
<th></th>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$E_3$</th>
<th>$E_4$</th>
<th>$E_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>day</strong> $t-1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>$\ell_{\leq t-1,i}$</strong></td>
<td><strong>1</strong></td>
<td><strong>2</strong></td>
<td><strong>1</strong></td>
<td><strong>2</strong></td>
<td><strong>3</strong></td>
</tr>
</tbody>
</table>
Main algorithms

<table>
<thead>
<tr>
<th>day ( t - 1 )</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ell_{\leq t-1,i} )</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

\[ E_1 \quad E_2 \quad E_3 \quad E_4 \quad E_5 \]

\[
\begin{array}{cccccc}
0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 \\
\end{array}
\]

\[
\hat{i}_t = \text{argmin}_i \ell_{\leq t-1,i} \quad \text{ties broken uniformly}
\]
Main algorithms

$$
\begin{array}{cccccc}
E_1 & E_2 & E_3 & E_4 & E_5 \\
0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 \\
\hline
day t - 1 & 0 & 0 & 1 & 1 & 1 \\
\hline
\ell_{\leq t-1,i} & 1 & 2 & 1 & 2 & 3
\end{array}
$$

FL \quad \hat{i}_t = \arg\min_i \ell_{\leq t-1,i} \quad \text{ties broken uniformly}

FPL(\eta) \quad \hat{i}_t = \arg\min_i \ell_{\leq t-1,i} + \frac{1}{\eta} \xi_{t,i} \quad \text{indep. additive noise}
Main algorithms

<table>
<thead>
<tr>
<th></th>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$E_3$</th>
<th>$E_4$</th>
<th>$E_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccccc}
\text{day } t - 1 & 0 & 0 & 1 & 1 & 1 \\
\ell_{\leq t-1,i} & 1 & 2 & 1 & 2 & 3 \\
\end{array}
\]

FL \quad \hat{i}_t = \arg\min_i \ell_{\leq t-1,i} \quad \text{ties broken uniformly}

FPL(\eta) \quad \hat{i}_t = \arg\min_i \ell_{\leq t-1,i} + \frac{1}{\eta} \xi_{t,i} \quad \text{indep. additive noise}

\text{Hedge}(\eta) \quad w_i = \frac{e^{-\eta \ell_{\leq t-1,i}}}{Z} \quad \text{soft min}
Dropout

\[
\begin{array}{cccccc}
E_1 & E_2 & E_3 & E_4 & E_5 \\
0 & \ne & 0 & 0 & \ne \\
1 & 1 & 0 & 1 & 1 \\
\text{day } t - 1 & 0 & 0 & \ne & \ne & 1 \\
\end{array}
\]

\[\hat{\ell}_{\leq t-1,i}\]
\[
\begin{array}{ccccc}
E_1 & E_2 & E_3 & E_4 & E_5 \\
0 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 \\
\end{array}
\]

day \( t - 1 \)

\[
\begin{array}{ccccc}
\hat{\ell}_{\leq t-1,i} & 1 & 1 & 0 & 1 & 2 \\
\end{array}
\]

\[
\hat{\ell}_{t,i} = \beta_{t,i} \ell_{t,i}, \quad \text{where } \beta_{t,i} \text{ iid Bernoulli}
\]
 Dropout

\[
\begin{array}{cccccc}
E_1 & E_2 & E_3 & E_4 & E_5 \\
0 & \checkmark & 0 & 0 & \checkmark \\
1 & 1 & 0 & 1 & 1 \\
\hline
day t - 1 & 0 & 0 & \checkmark & \checkmark & 1 \\
\end{array}
\]

\[\hat{\ell}_{\leq t-1,i} = 1 \quad 1 \quad 0 \quad 1 \quad 2\]

\[\hat{\ell}_{t,i} = \beta_{t,i} \ell_{t,i}, \quad \text{where } \beta_{t,i} \text{ iid Bernoulli}\]

\[\hat{i}_t = \arg\min_i \hat{\ell}_{\leq t-1,i}\]

FL on dropout
How good?

Optimal worst case regret: $\sqrt{L^* \ln n} + \ln n$
How good?

Optimal worst case regret: $\sqrt{L^* \ln n} + \ln n$

- FL is bad
- FPL($\eta$) and Hedge($\eta$) achieve optimal regret with tuning
  - fancy tunings: AdaHedge and Flipflop
How good?

Optimal worst case regret: $\sqrt{L^* \ln n + \ln n}$

- FL is bad
- FPL($\eta$) and Hedge($\eta$) achieve optimal regret with tuning
  - fancy tunings: AdaHedge and Flipflop
- FL on dropout requires no tuning
How good?

Optimal worst case regret: $\sqrt{L^* \ln n} + \ln n$

- FL is bad
- FPL($\eta$) and Hedge($\eta$) achieve optimal regret with tuning
  - fancy tunings: AdaHedge and Flipflop
- FL on dropout requires no tuning
  - dropout better noise for achieving optimal worst case regret
  - additive noise needs tuning - multiplicative noise does not
- in iid case when gap between 1st and 2nd: $\log n$ regret
How good?

Optimal worst case regret: $\sqrt{L^* \ln n} + \ln n$

- FL is bad
- FPL($\eta$) and Hedge($\eta$) achieve optimal regret with tuning
  - fancy tunings: AdaHedge and Flipflop
- FL on dropout requires no tuning
  - dropout better noise for achieving optimal worst case regret
    - additive noise needs tuning - multiplicative noise does not
  - in iid case when gap between 1st and 2nd: $\log n$ regret
- In the meantime
  - new fancy algorithms by Haipeng Luo, Rob Schapire & Tim van Erven, Wouter Koolen
How good?

Optimal worst case regret: \( \sqrt{L^* \ln n + \ln n} \)

- FL is bad
- FPL(\(\eta\)) and Hedge(\(\eta\)) achieve optimal regret with tuning
  - fancy tunings: AdaHedge and Flipflop
- FL on dropout requires no tuning
  - dropout better noise for achieving optimal worst case regret
  - additive noise needs tuning - multiplicative noise does not
- in iid case when gap between 1st and 2nd: \(\log n\) regret

In the meantime
- new fancy algorithms by Haipeng Luo, Rob Schapire & Tim van Erven, Wouter Koolen
- also no tuning, many other advantages
- but more complicated
What regularization?

Hedge(\(\eta\)) relative entropy
What regularization?

\[ \text{Hedge}(\eta) \quad \text{relative entropy} \]
\[ \text{FPL}(\eta) \quad \text{additive } \frac{1}{\eta} \log \text{exponential noise} = \text{Hedge}(\eta) \]
What regularization?

Hedge($\eta$)  relative entropy
FPL($\eta$)  additive $\frac{1}{\eta}$ log exponential noise = Hedge($\eta$)

FL on dropout  tricky

Feed forward NN  [Wagner, Wang, Liang 2013]
Logistic regression  [Helmbold, Long 2014]
Linear loss case  [ALST 2014]
Any deterministic alg. (such as FL) has huge regret

- For \( T \) trials: give algorithm’s expert a unit of loss
- Loss of alg.: \( T \) \( \frac{T}{n} \)
loss of best: \( \leq \frac{T}{n} \)
Any deterministic alg. (such as FL) has huge regret

- For $T$ trials: give algorithm’s expert a unit of loss
- Loss of alg.: $T$ loss of best: $\leq \frac{T}{n}$

$$ \text{regret: } \geq \frac{T}{nL^*} - \frac{T}{n} = (n - 1)L^* $$
Any deterministic alg. (such as FL) has huge regret

- For $T$ trials: give algorithm’s expert a unit of loss
- Loss of alg.: $T$  
  loss of best: $\leq \frac{T}{n}$
  
  \[
  \text{regret: } \geq \frac{T}{nL^*} - \frac{T}{L^*} = (n - 1)L^*
  \]

Recall optimum regret: $\sqrt{L^* \ln n} + \ln n$

FL with random ties

- Loss of alg.: $(L^* + 1) \ln n$  
  loss of best: $L^*$
  
  regret: $L^* \ln n$
Our analysis of dropout

Unit rule
- Adversary forces more regret by splitting loss vectors into units

\[
\begin{pmatrix}
1 \\
0 \\
1 \\
1 \\
\end{pmatrix}
\rightarrow
\begin{pmatrix}
1 \\
0 \\
0 \\
1 \\
\end{pmatrix}
\]
Our analysis of dropout

**Unit rule**

- Adversary forces more regret by splitting loss vectors into units

\[
\begin{pmatrix}
1 \\
0 \\
1 \\
1
\end{pmatrix} \rightarrow \begin{pmatrix}
1 \\
0 \\
0 \\
1
\end{pmatrix}, \begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}
\]

**Swapping rule**

\[
\ell_{\leq T,i}
\]

| \(E_1\) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 9  |
| \(E_2\) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8  |
| \(E_3\) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6  |
| \(E_4\) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6  |
Our analysis of dropout

Unit rule

- Adversary forces more regret by splitting loss vectors into units

\[
\begin{pmatrix}
1 \\
0 \\
1 \\
1
\end{pmatrix}
\rightarrow
\begin{pmatrix}
1 \\
0 \\
0 \\
1
\end{pmatrix}
\begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}
\begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}
\]

Swapping rule

\[\ell_{\leq T,i}\]

| \(E_1\) | 1 1 1 1 1 1 1 1 1 1 | 9 |
| \(E_2\) | 1 1 1 1 1 1 1 1 1 | 8 |
| \(E_3\) | 1 1 1 1 1 1 1 1 1 1 | 10 |
| \(E_4\) | 1 1 1 1 1 1 1 | 6 |

- 1’s occur in some order
- Worst case: 1 before 1
- Otherwise adversary benefits from swapping
Worst-case pattern

1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
Assume we have $s$ leaders
Cost per sweep

Assume we have $s$ leaders

$s$ leader get unit
ignore non-leaders

\[
\begin{pmatrix}
1 \\
1 \\
1 \\
1 \\
1
\end{pmatrix}
\]
Cost per sweep

Assume we have $s$ leaders

- $s$ leader get unit
- ignore non-leaders

\[
\begin{array}{c}
\text{FL (with random ties)} \\
\frac{1}{s} + \frac{1}{s-1} + \frac{1}{s-2} + \frac{1}{s-3} + \ldots + \frac{1}{s-s-2} + \frac{1}{s-s-1}
\end{array}
\]

\[\approx \ln s\]
Assume we have $s$ leaders

\[
\begin{aligned}
&\begin{cases}
1 \\
1 \\
1 \\
1 \\
1
\end{cases} \\
&\begin{cases}
1 \\
1 \\
1 \\
1 \\
1
\end{cases} \\
&\begin{cases}
1 \\
1 \\
1 \\
1 \\
1
\end{cases}
\end{aligned}
\]

**FL (with random ties)**

\[
\frac{1}{s} + \frac{1}{s-1} + \frac{1}{s-2} + \frac{1}{s-3} + \ldots + \frac{1}{s-s-2} + \frac{1}{s-s-1}
\]

\[ \approx \ln s \]

**Dropout**

\[
\frac{1}{s} + \frac{1}{s-1/2} + \frac{1}{s-2/2} + \frac{1}{s-3/2} + \ldots + \frac{1}{s-(s-2)/2} + \frac{1}{s-(s-1)/2}
\]

\[ \approx 2 \ln \frac{2s}{s} = 2 \ln 2 \]
Overview of proof for noisy case

- Focus on first $L$ sweeps
- Only occurs constant regret if number of leaders $> 1$
Overview of proof for noisy case

- Focus on first $L$ sweeps
- Only occurs constant regret if number of leaders > 1
- Prob. that number of leaders > 1 is at most $\sqrt{\frac{\ln n}{q+1}}$ for sweep $q$
- In first $L^*$ sweeps $O(\sqrt{L^* \ln n})$ regret
- Note much more loss in runaway phase
Overview of proof for noisy case

- Focus on first $L$ sweeps
- Only occurs constant regret if number of leaders $> 1$

- Prob. that number of leaders $> 1$ is at most $\sqrt{\frac{\ln n}{q+1}}$ for sweep $q$

- In first $L^*$ sweeps $O(\sqrt{L^* \ln n})$ regret
- Note much more loss in runaway phase

- For Hedge($\eta$) and FPL($\eta$) cost per sweep constant and dependent on $\eta$
Dropout versus Hedge

![Graph showing regret vs sweep t for Dropout and tuned Hedge](image-url)
Outlook

- Combinatorial experts
- Matrix case
- Where else can dropout perturbations be used?
- Dropout for convex losses
- Dropout for neural nets
Outlook

- Combinatorial experts
- Matrix case
- Where else can dropout perturbations be used?
- Dropout for convex losses
- Dropout for neural nets
- Privacy