Online Linear Learning

John Langford, Machine Learning the Future, February 27

To follow along:

git clone
git://github.com/JohnLangford/vowpal_wabbit.git
wget http://hunch.net/~jl/rcv1.tar.gz
Features: a vector $x \in \mathbb{R}^n$
Label: $y \in \mathbb{R}$
Goal: Learn $w \in \mathbb{R}^n$ such that
$\hat{y}_w(x) = \sum_i w_i x_i$ is close to $y$. 

Features: a vector $\mathbf{x} \in \mathbb{R}^n$
Label: $y \in \{-1, 1\}$
Goal: Learn $\mathbf{w} \in \mathbb{R}^n$ such that
$\hat{y}_w(\mathbf{x}) = \text{sign}(\sum_i w_i x_i) = y$. 
Online Linear Learning

Start with $\forall i : w_i = 0$
Repeatedly:

1. Get features $x \in \mathbb{R}^n$.
2. Make linear prediction $\hat{y}_w(x) = \sum_i w_i x_i$.
3. Observe label $y \in [0, 1]$.
4. Update weights so $\hat{y}_w(x)$ is closer to $y$. 

Example: $w_i \leftarrow w_i + \eta (y - \hat{y}_w)$.
Online Linear Learning

Start with $\forall i : \ w_i = 0$

Repeatedly:

1. Get features $x \in \mathbb{R}^n$.
2. Make logistic prediction $\hat{y}_w(x) = \frac{1}{1 + e^{-\sum_i w_i x_i}}$.
3. Observe label $y \in [0, 1]$.
4. Update weights so $\hat{y}_w(x)$ is closer to $y$. 

Example: $w_i \leftarrow w_i + \eta (y - \hat{y}_w)$.
Online Linear Learning

Start with $\forall i : w_i = 0$
Repeatedly:

1. Get features $x \in \mathbb{R}^n$.
2. Make linear prediction $\hat{y}_w(x) = \sum_i w_i x_i$.
3. Observe label $y \in [0, 1]$.
4. Update weights so $\hat{y}_w(x)$ is closer to $y$. 

Example: $w_i \leftarrow w_i + \eta (y - \hat{y}_w) x_i$. 
Online Linear Learning

Start with $\forall i : \ w_i = 0$
Repeatedly:

1. Get features $x \in \mathbb{R}^n$.
2. Make linear prediction $\hat{y}_w(x) = \sum_i w_i x_i$.
3. Observe label $y \in [0, 1]$.
4. Update weights so $\hat{y}_w(x)$ is closer to $y$.
   Example: $w_i \leftarrow w_i + \eta(y - \hat{y})x_i$. 
An Example: The RCV1 dataset

Pick whether a document is in category CCAT or not.

Dataset size:

- **781K** examples
- **60M** nonzero features
- **1.1G** bytes

Format: 

```
label | sparse features ...
```

command: 

```
time vw --sgd rcv1.train.txt -c
```

takes 1-3 seconds on my laptop.
An Example: The RCV1 dataset

Pick whether a document is in category CCAT or not.
Dataset size:
781K examples
60M nonzero features
1.1G bytes
Format: label | sparse features ...
1 | 13:3.9656971e-02 24:3.4781646e-02 ...
An Example: The RCV1 dataset

Pick whether a document is in category CCAT or not.

Dataset size:
781K examples
60M nonzero features
1.1G bytes

Format: label | sparse features ...
1 | 13:3.9656971e-02 24:3.4781646e-02 ...
which corresponds to:
1 | tuesday year ...
An Example: The RCV1 dataset

Pick whether a document is in category CCAT or not.

Dataset size:
781K examples
60M nonzero features
1.1G bytes

Format: label | sparse features ...
1 | 13:3.9656971e-02 24:3.4781646e-02 ...

which corresponds to:
1 | tuesday year ...

command: time vw --sgd rcv1.train.txt -c
takes 1-3 seconds on my laptop.
Reasons for Online Learning

1. Fast convergence to a good predictor
2. It’s RAM efficient. You need store only one example in RAM rather than all of them. ⇒ Entirely new scales of data are possible.
3. Online Learning algorithm = Online Optimization Algorithm. Online Learning Algorithms ⇒ the ability to solve entirely new categories of applications.
4. Online Learning = ability to deal with drifting distributions.
Defining updates

1 Define a loss function $L(\hat{y}_w(x), y)$.

2 Update according to $w_i \leftarrow w_i - \eta \frac{\partial L(\hat{y}_w(x), y)}{\partial w_i}$.

Here $\eta$ is the learning rate.
Defining updates

1. Define a loss function $L(\hat{y}_w(x), y)$. 

Here $\eta$ is the learning rate.
Defining updates

1. Define a loss function $L(\hat{y}_w(x), y)$.
2. Update according to $w_i \leftarrow w_i - \eta \frac{\partial L(\hat{y}_w(x), y)}{\partial w_i}$.

Here $\eta$ is the learning rate.
Defining updates

1. Define a loss function $L(\hat{y}_w(x), y)$.

2. Update according to $w_i \leftarrow w_i - \eta \frac{\partial L(\hat{y}_w(x), y)}{\partial w_i}$.

Here $\eta$ is the learning rate.

common loss functions

- 0/1
- squared
- logistic
- quantile
- hinge
Know your loss function semantics

1. What is a typical price for a house?

2. What is the expected return on a stock?

3. What is the probability of a click on an ad?

4. Is the digit a 1?

5. What do you really care about?
Know your loss function semantics

1. What is a typical price for a house?
   quantile: minimizer = median

2. What is the expected return on a stock?
   squared: minimizer = expectation

3. What is the probability of a click on an ad?
   logistic: minimizer = probability

4. Is the digit a 1?
   hinge: closest 0/1 approximation

5. What do you really care about?
   often 0/1
A proof for quantile regression

Consider conditional probability distribution $D(y|x)$. 

![Probability Distribution](image-url)
A proof for quantile regression

Consider equal mass tails. Where is loss minimized?
A proof for quantile regression

Minimizer is always between.
A proof for quantile regression

Works for any tails $\Rightarrow$ works for mass 0.5 tails.
How do you know when you succeed?

Progressive Validation

On timestep $t$, let $l_t = L(\hat{y}_w(x_t), y_t)$.

Report loss $L = \mathbb{E}_t l_t$.

PV analysis

Let $D$ be a distribution over $x, y$. Let $\bar{l}_t = \mathbb{E}_{(x, y) \sim D} L(\hat{y}_w(x), y)$.

Theorem: For all probability distributions $D(x, y)$, for all online learning algorithms, with probability $1 - \delta$:

$$\left| L - \mathbb{E}_t \bar{l}_t \right| \leq \sqrt{\frac{\ln 2}{\delta^2 T}}$$
How do you know when you succeed?

Progressive Validation

On timestep $t$ let $l_t = L(\hat{y}_{wt}(x_t), y_t)$.
Report loss $L = E_t l_t$. 
Progressive Validation

On timestep $t$ let $l_t = L(\hat{y}_{wt}(x_t), y_t)$.
Report loss $L = E_t l_t$.

PV analysis

Let $D$ be a distribution over $x, y$. Let $\bar{l}_t = E_{(x,y) \sim D} L(\hat{y}_{wt}(x), y)$
Theorem: For all probability distributions $D(x,y)$, for all online learning algorithms, with probability $1 - \delta$:

$$|L - E_t \bar{l}_t| \leq \sqrt{\frac{\ln 2/\delta}{2T}}$$
All the common loss functions are sound for binary classification, so which is best is an empirical choice.
vw --sgd rcv1.train.txt -c --loss_function hinge --binary
vw --sgd rcv1.train.txt -c --loss_function logistic --binary
vw --sgd rcv1.train.txt -c --loss_function quantile --binary
All the common loss functions are sound for binary classification, so which is best is an empirical choice.

```
vw --sgd rcv1.train.txt -c --loss_function hinge --binary
vw --sgd rcv1.train.txt -c --loss_function logistic --binary
vw --sgd rcv1.train.txt -c --loss_function quantile --binary
```

Progressive validation often does not replace train/test discipline, but it can greatly aid empirical testing.
Part II, advanced updates

1. Importance weight invariance
2. Adaptive updates
3. Normalized updates
A common scenario: you need to do classification but one choice is more expensive than the other.

An example: In spam detection, predicting nonspam as spam is worse than spam as nonspam.
Learning with importance weights

A common scenario: you need to do classification but one choice is more expensive than the other.

An example: In spam detection, predicting nonspam as spam is worse than spam as nonspam.

Let’s say an example is $I$ times more important than a typical example. How do you modify the update to use $I$?
A common scenario: you need to do classification but one choice is more expensive than the other.

An example: In spam detection, predicting nonspam as spam is worse than spam as nonspam.

Let’s say an example is $I$ times more important than a typical example. How do you modify the update to use $I$?

The baseline approach: $w_i \leftarrow w_i - \eta I \frac{\partial L(\hat{y}_w(x), y)}{\partial w_i}$. 
Dealing with the importance weights

\[ w_i \leftarrow w_i - \eta I \frac{\partial L(\hat{y}_w(x), y)}{\partial w_i} \]

performs poorly.

![Graph showing Baseline Importance Update and Baseline Update](image-url)
Dealing with the importance weights

A better approach: \( w_i \leftarrow w_i - \eta \frac{\partial L(\hat{y}_w(x), y)}{\partial w_i} \) \( I \) times

Baseline Importance Update

Square Loss
Baseline Update
Repeated update

Baseline Importance Update

loss

prediction when \( y=1 \)
Dealing with the importance weights

An even better approach: \( w_i \leftarrow w_i - s(\eta I) \frac{\partial L(\hat{y}_w(x), y)}{\partial w_i} \)

![Graph showing Baseline Importance Update with different loss functions and predictions when \( y = 1 \).]
Robust results for unweighted problems

- Astro - logistic loss
- Spam - quantile loss
- RCV1 - squared loss
- Webspam - hinge loss
rcv1 with an invariant update

```
vw rcv1.train.txt -c --binary --invariant
```
Performs slightly worse with the default learning rate, but much more robust to learning rate choice.
Adaptive Learning

Learning rates must decay to converge, but how?
Adaptive Learning

Learning rates must decay to converge, but how?

Common answer: \( \eta_t = 1/t^{0.5} \) or \( \eta_t = 1/t \).
Adaptive Learning

Learning rates must decay to converge, but how?
Common answer: $\eta_t = 1/t^{0.5}$ or $\eta_t = 1/t$.

Better answer: $t$, let $g_{it} = \frac{\partial L(\hat{y}_w(x_t), y_t)}{\partial w_i}$.
New update rule: $w_i \leftarrow w_{it} - \eta \frac{g_{it}}{\sqrt{\sum_{t'=1}^t g_{it'}^2}}$. 
Common features stabilize quickly. Rare features can have large updates.
Adaptive Learning

Learning rates must decay to converge, but how? Common answer: $\eta_t = 1/t^{0.5}$ or $\eta_t = 1/t$.

Better answer: $t$, let $g_{it} = \frac{\partial L(\hat{y}_w(x_t), y_t)}{\partial w_i}$.
New update rule: $w_i \leftarrow w_{it} - \eta \frac{g_{it}}{\sqrt{\sum_{t'=1}^{t} g_{it'}^2}}$

Common features stabilize quickly. Rare features can have large updates.
Adaptive Learning example

vw rcv1.train.txt -c --binary --adaptive
Slightly worse. Adding in --invariant -l 1 helps.
Dimensional Correction

\[ g_{it} \text{ for squared loss } = 2(\hat{y}_w(x) - y)x_i \text{ so update is } \\
\]

\[ w_i \leftarrow w_i - Cx_i \]

The same form occurs for all linear updates.
Dimensional Correction

\( g_{it} \) for squared loss is \( 2(\hat{y}_w(x) - y)x_i \) so update is

\[
    w_i \leftarrow w_i - Cx_i
\]

The same form occurs for all linear updates. Intrinsic problems! Doubling \( x_i \) implies halving \( w_i \) to get the same prediction.

\( \Rightarrow \) Update rule has mixed units!
For each feature $x_i$ compute:

- empirical mean $\mu_i = E_t x_{it}$
- empirical standard deviation $\sigma_i = \sqrt{E_t (x_{it} - \mu_i)^2}$

Let $x_i' \leftarrow \frac{x_i - \mu_i}{\sigma_i}$. 

A standard solution: Gaussian sphering
A standard solution: Gaussian sphering

For each feature $x_i$ compute:

empirical mean $\mu_i = E_t x_{it}$

empirical standard deviation $\sigma_i = \sqrt{E_t (x_{it} - \mu_i)^2}$

Let $x_i' \leftarrow \frac{x_i - \mu_i}{\sigma_i}$.

Problems:

1. Lose online.
2. RCV1 becomes a factor of 500 larger.
A scale-free update

NG(learning_rate $\eta$)

1. Initially $w_i = 0$, $s_i = 0$, $N = 0$
2. For each timestep $t$ observe example $(x, y)$
   1. For each $i$, if $|x_i| > s_i$
      1. Renormalize $w_i$ for new scale
      2. Adjust Scale
   2. $\hat{y} = \sum_i w_i x_i$
   3. Adjust global scale
   4. For each $i$,
      1. $w_i \leftarrow w_i - \eta$ (scale adjustment) $\frac{\partial L(\hat{y}, y)}{\partial w_i}$
A scale-free update

NG(learning rate $\eta$)

1. Initially $w_i = 0$, $s_i = 0$, $N = 0$
2. For each timestep $t$ observe example $(x, y)$
   1. For each $i$, if $|x_i| > s_i$
      1. $w_i \leftarrow \frac{w_is_i^2}{|x_i|^2}$
      2. Adjust Scale
   2. $\hat{y} = \sum_i w_i x_i$
3. Adjust global scale
4. For each $i$,
   1. $w_i \leftarrow w_i - \eta \text{ (scale adjustment)} \frac{\partial L(\hat{y}, y)}{\partial w_i}$
A scale-free update

NG(learning_rate $\eta$)

1. Initially $w_i = 0$, $s_i = 0$, $N = 0$
2. For each timestep $t$ observe example $(x, y)$
   1. For each $i$, if $|x_i| > s_i$
      1. $w_i \leftarrow \frac{w_i s_i^2}{|x_i|^2}$
      2. $s_i \leftarrow |x_i|$
   2. $\hat{y} = \sum_i w_i x_i$
3. Adjust global scale
4. For each $i$,
   1. $w_i \leftarrow w_i - \eta$ (scale adjustment) $\frac{\partial L(\hat{y}, y)}{\partial w_i}$
A scale-free update

NG(learning_rate $\eta$)

1. Initially $w_i = 0$, $s_i = 0$, $N = 0$
2. For each timestep $t$ observe example $(x, y)$
   1. For each $i$, if $|x_i| > s_i$
      1. $w_i \leftarrow \frac{w_i s_i^2}{|x_i|^2}$
      2. $s_i \leftarrow |x_i|$
   2. $\hat{y} = \sum_i w_i x_i$
   3. $N \leftarrow N + \sum_i \frac{x_i^2}{s_i^2}$
   4. For each $i$,
      1. $w_i \leftarrow w_i - \eta$ (scale adjustment) $\frac{\partial L(\hat{y}, y)}{\partial w_i}$
A scale-free update

NG(learning_rate $\eta$)

1. Initially $w_i = 0$, $s_i = 0$, $N = 0$

2. For each timestep $t$ observe example $(x, y)$
   
   1. For each $i$, if $|x_i| > s_i$
      
      1. $w_i \leftarrow \frac{w_is_i^2}{|x_i|^2}$
      2. $s_i \leftarrow |x_i|$

   2. $\hat{y} = \sum_i w_i x_i$

   3. $N \leftarrow N + \sum_i \frac{x_i^2}{s_i^2}$

   4. For each $i$,
      
      1. $w_i \leftarrow w_i - \eta \sqrt{\frac{t}{N} \frac{1}{s_i^2}} \frac{\partial L(\hat{y}, y)}{\partial w_i}$
In combination

An adaptive, scale-free, importance invariant update rule.

`vw rcv1.train.txt -c --binary`
... there are many more problems with gradient descent. How do you fix them?
References


References


[Importance Aware Updates] Nikos Karampatziakis and John Langford, Importance Weight Aware Gradient Updates UAI 2010.

