Online Importance Weight Aware Updates

Nikos Karampatziakis[*]
Department of Computer Science
Cornell University
nk@cs.cornell.edu

John Langford
Yahoo! Research
jl@yahoo-inc.com

Abstract

An importance weight quantifies the relative importance of one example over another, coming up in applications of boosting, asymmetric classification costs, reductions, and active learning. The standard approach for dealing with importance weights in gradient descent is via multiplication of the gradient. We first demonstrate the problems of this approach when importance weights are large, and argue in favor of more sophisticated ways for dealing with them. We then develop an approach which enjoys an invariance property: that updating twice with importance weight h is equivalent to updating once with importance weight $2h$. For many important losses this has a closed form update which satisfies standard regret guarantees when all examples have $h = 1$. We also briefly discuss two other reasonable approaches for handling large importance weights. Empirically, these approaches yield substantially superior prediction with similar computational performance while reducing the sensitivity of the algorithm to the exact setting of the learning rate. We apply these to online active learning yielding an extraordinarily fast active learning algorithm that works even in the presence of adversarial noise.

1 INTRODUCTION

Importance weights appear in boosting algorithms [9] which assign a weight to each example depending on how well this point has been classified in previous iterations, covariate shift algorithms [11] which assign a weight to a training example according to how close to the test distribution the example is, and active learning algorithms [1, 2] where an adaptive rejection sampling scheme is applied to each example and each retained example gets an importance equal to the inverse probability of being retained. Importance weights have become a de-facto language for specifying the relative importance of prediction amongst examples.

When not concerned by computational constraints, importance weights can be dealt with using either black box techniques [19, 20] or direct modification of existing algorithms such that an existing example with importance weight h is treated as h examples. However, when computational constraints are significant online gradient descent based algorithms are preferred. Here the standard approach of treating an example with importance weight h as h examples is typically translated into practice via multiplying the gradient by h. This is undesirable for large h because such an example can cause an update that's far beyond what's necessary to attain a small loss on it.

An important observation is that multiplying the gradient by h is typically *not* equivalent to doing h updates via gradient descent because all loss functions of interest are nonlinear. The goal of this paper is resolving this translation failure by investigating alternate updates that gracefully deal with importance weights, by taking into account the curvature of the loss. Among these updates we mainly focus on a novel set of updates that satisfies an additional *invariance* property: for all importance weights of h, the update is equivalent to two updates with importance weight $h/2$. We call these updates *importance invariant.*

Even though the importance invariant updates will be defined via an ordinary differential equation (ODE), we were surprised to find that they are closed-form for all common loss functions. We were also surprised to discover that the importance weight invariant update substantially improves the learned predictor even when $h = 1$, both in terms of the quality of best pre-

[*]Research done while author was at Yahoo! Research

dictor after a parameter search and in terms of the robustness to parameter search, effectively reducing the desirability of searching over some schedule of learning rates. Upon inspection, the reason for this is that an importance weight invariant update smoothly interpolates between a very aggressive projection [10, 5] algorithm and a less aggressive gradient multiplier decay algorithm. All of these benefits come at near-zero computational cost.

Among the other algorithms we consider, implicit updates [14, 15] turn out to coincide with importance invariant ones for piecewise linear losses and provide qualitatively similar updates for other losses, implying that our derivation is an alternative way to motivate this style of algorithm. For most other loss functions implicit updates require a root-finding algorithm.

Finally, another reasonable way to handle importance is related to [6], who analyze it for the logistic and exponential losses. Here, a second order Taylor expansion at the current prediction and in the direction of the update is used to approximate the loss. These updates coincide with implicit updates for squared loss (since the quadratic approximation is exact) and are not applicable to piecewise linear losses. We won't discuss these updates any further since they have only been analyzed for very specific loss functions.

In section 2 we define the problem and describe some obvious but unsatisfactory approaches. Next, we propose the importance invariant solution and present a general framework for deriving importance invariant updates for many loss functions, in section 3. We subsequently discuss some important properties of the proposed updates, such as safety, in section 4. Section 5 briefly covers how implicit updates can handle importance weights. In section 6 we empirically demonstrate the merits of not linearizing the loss on problems with and without importance weights. Section 7 states our conclusions.

2 PROBLEM SETTING

We assume access to a training set of triplets (x_t, y_t, h_t), $t = 1, \ldots, T$ where $x_t \in \mathbb{R}^d$ is a vector of d features, $h_t \in \mathbb{R}_+$ is an importance weight, and $y_t \in \mathbb{R}$ is a label. We are also given a loss function $\ell(p, y)$ where p is the prediction of our model and y is the actual label. Depending on the loss function, y may take values in a restricted set, such as $\{-1, +1\}$ or $\{0, 1\}$. In this paper we focus on linear models i.e. $p = w^\top x$ where $w \in \mathbb{R}^d$ is a vector of weights. Our goal is to find

$$w = \operatorname*{argmin}_w \sum_{t=1}^{T} h_t \ell(w^\top x_t, y_t), \qquad (1)$$

Algorithm 1 Online Gradient Descent

```
$w_1 \leftarrow 0$
**for** $t = 1$ to $T$ **do**
    $w_{t+1} \leftarrow w_t - \eta_t \nabla_w \ell(w_t^\top x_t, y_t)$
**done**
```

using online gradient descent. When examples do *not* have importance weights the online gradient descent algorithm is shown in Algorithm 1. The notation $\nabla_w \ell(w_t^\top x_t, y_t)$ means the gradient of the loss with respect to w evaluated at the t-th prediction and label.

When examples have importance weights, we would like to adhere to the following principle: *An example with importance weight h should be treated as if it is a regular example that appears h times in the dataset.* This is a statement of both mathematical and semantic correctness. Mathematically, (1) states exactly the same thing. Semantically an example of importance h is just a convenient encoding of h identical examples. For now we assume importance weights are integers and the learning rate sequence is constant $\eta_t = \eta$. These assumptions are only for ease of exposition and are lifted in section 3.

2.1 Some Unsatisfactory Approaches

A first approach would be to loop through the data enough epochs, with epoch i using only those examples whose importance is greater than i. While this is a valid approach, it is very inefficient. Ideally each example should be presented once to the learner.

Another tempting approach is multiplying the update by the importance weight:

$$w_{t+1} = w_t - h_t \eta \nabla_w \ell(w_t^\top x_t, y_t).$$

However, this update rule does not respect the principle of the previous section. To see this, consider the case $h_t = 2$. The above rule should be equivalent to

$$\begin{aligned} v &= w_t - \eta \nabla_w \ell(w_t^\top x_t, y_t) \\ w_{t+1} &= v - \eta \nabla_w \ell(v^\top x_t, y_t) \end{aligned}$$

which is not true in general. Furthermore, the quality of this update gets worse as the importance weight gets larger since the first order approximation of the loss is invalid far away from its expansion point.

Another approach with good computational characteristics is rejection sampling accoding to $h/h_{\max}$. However, this approach generally decreases performance due to throwing out samples. Rejection sampling can be repaired by learning multiple predictors based upon different rejection sampled datasets [20], but this of course increases computation substantially.

2.2 An Efficient Invariant Approach

To achieve invariance and efficiency we will focus on the cumulative effect of presenting an example h times in a row. This scheme respects our correctness principle and considers each example once. The only remaining question is whether the cumulative effect of h presentations in a row can be computed faster than explicitly doing so. In the next section we explain why this is possible and how to compute it.

3 A FRAMEWORK FOR DERIVING STEP SIZES

Given a loss of the form $\ell(p, y)$ where p is the prediction and assuming a linear model $p = w^\top x$ we have that $\nabla_w \ell = \frac{\partial \ell}{\partial p} x$. Therefore all gradients of a given example point to the same direction and only differ in magnitude. Hence computing the cumulative effect of presenting the example h times in a row amounts to computing a global scaling for x that aggregates the effects of all the gradients. We begin with a simple lemma that formalizes this in the case of integer importance weights.

Lemma 1. *Let $h \in \mathbb{N}$. Presenting example (x, y) h times in a row is equivalent to the update*

$$w_{t+1} = w_t - s(h)x \tag{2}$$

where the scaling factor $s(h)$ has this recursive form:

$$s(h+1) = s(h) + \eta \left.\frac{\partial \ell}{\partial p}\right|_{p=(w_t - s(h)x)^\top x} \tag{3}$$

$$s(0) = 0 \tag{4}$$

Proof. By induction on h. The base case is obvious. Now the effect of presenting the example (x, y) $h+1$ times can be computed by performing a gradient update on the vector v that results from presenting the example h times. By the induction hypothesis this intermediate vector is

$$v = w_t - s(h)x$$

and the gradient descent step is

$$w_{t+1} = v - \eta \nabla_w \ell(w^\top x, y)|_{w=v}.$$

Expanding this using the induction hypothesis we get

$$\begin{aligned} w_{t+1} &= w_t - s(h)x - \eta \left.\frac{\partial \ell}{\partial p}\right|_{p=v^\top x} x \\ &= w_t - \left(s(h) + \eta \left.\frac{\partial \ell}{\partial p}\right|_{p=(w_t - s(h)x)^\top x} \right) x \end{aligned}$$

$\square$

Given a loss function, one could try to find a closed form solution to the recurrence defined by (3),(4). For example for squared loss $\ell(p, y) = \frac{1}{2}(p-y)^2$ the recurrence is

$$s(h+1) = s(h) + \eta((w_t - s(h)x)^\top x - y).$$

A simple inductive argument can then verify that

$$s(h) = \frac{w_t^\top x - y}{x^\top x}(1 - (1 - \eta x^\top x)^h) \tag{5}$$

Note that when $\eta x^\top x < 1$, $s(h)$ asymptotes to the quantity that would make $w_{t+1}^\top x = y$. This behavior is more desirable than that of multiplying the gradient with the importance weight.

Notwithstanding the significance of such an update, it is restricted to integer importance weights. Moreover other loss functions do not yield a recurrence with a closed form solution. To overcome these problems, we use (5) as a starting point to think about the consequences of presenting an example many times. To compensate, we will also need to adjust the learning rate we use everytime we present an example. Suppose that we present an example a factor of n times more using a learning rate that is smaller by a factor of n. This can be simulated in constant time using (5) with hn and $\frac{\eta}{n}$ in place of h and η respectively. Letting n grow large, we are interested in

$$\lim_{n \to \infty} \frac{w_t^\top x - y}{x^\top x} \left(1 - \left(1 - \frac{\eta x^\top x}{n} \right)^{nh} \right)$$

Using that $\lim_{n \to \infty}(1 + z/n)^n = e^z$ we have that

$$s(h) = \frac{w_t^\top x - y}{x^\top x} \left(1 - \exp(-h\eta x^\top x) \right). \tag{6}$$

The key in the above derivation is using the limit of the gradient descent process as the learning rate becomes infinitesimal. We now generalize this idea to derive updates for other loss functions.

Theorem 1. *The limit of the gradient descent process as the learning rate becomes infinitesimal for an example with importance weight $h \in \mathbb{R}_+$ is equal to the update*

$$w_{t+1} = w_t - s(h)x$$

where the scaling factor $s(h)$ satisfies the differential equation:

$$s'(h) = \eta \left.\frac{\partial \ell}{\partial p}\right|_{p=(w_t - s(h)x)^\top x}, \quad s(0) = 0 \tag{7}$$

The proof is in the full version of the paper [13]. This theorem is our framework for deriving updates for many loss functions. Plugging a loss function in (7) gives an ODE whose solution is the result of a continuous gradient descent process. The ODE can be easily solved by separation of variables.

As a sanity check for squared loss, $\frac{\partial \ell}{\partial p} = p - y$, (7) gives

$$s'(h) = \eta((w_t - s(h)x)^\top x - y), \quad s(0) = 0$$

a linear ODE, whose solution exactly rederives (6).

3.1 Other Loss Functions

Using (7) as our framework, we can derive step sizes for many popular loss function as summarized in table 1.

For the logistic loss, the solution involves the Lambert W function: $W(z)e^{W(z)} = z$, and the solution can be verified using $W'(z) = \frac{W(z)}{z(1+W(z))}$. The exponential loss also fits nicely into our framework.

For the logarithmic loss the ODE has no explicit form for all $y \in [0, 1]$. The table presents the common case $y \in \{0, 1\}$. In this case each value of y gives rise to an ODE whose solution has an explicit form. Note that here the ODE solutions satisfy a second degree equation and hence each branch has two solutions. We have selected the one satisfying $s'(0) = \eta \frac{\partial \ell}{\partial p}$. To avoid an infinite loss, we can clip the predictions away from 0 and 1 (and update using $\min(h, h')$ where h' is the importance weight that hits the clipping point) or use a link function such as tanh. In the full version [13] we show updates for this latter case.

A similar situation arises for the Hellinger loss. The solution to (7) has no simple form for all $y \in [0, 1]$ but for $y \in \{0, 1\}$ we get the expressions in table 1.

3.1.1 Hinge Loss and Quantile Loss

Two other commonly used loss function are the hinge loss and the τ-quantile loss where $\tau \in [0, 1]$ is a parameter. These are differentiable everywhere except at one point where the subdifferential contains zero.

Hence, for the hinge loss, a valid expression for (7) is

$$s'(h) = \begin{cases} -\eta y & y(w - s(h)x)^\top x < 1 \\ 0 & y(w - s(h)x)^\top x \geq 1 \end{cases}$$

The first branch (together with $s(0) = 0$) gives $s(h) = -yh\eta$ for $y(w + yh\eta x)^\top x < 1$. Otherwise, i.e. when $h \geq h_{\text{hinge}} = \frac{1 - yw^\top x}{\eta x^\top x}$, $s(h)$ is a constant. Here h_{hinge} is the importance weight that would make the updated prediction lie at the hinge. To maintain continuity at h_{hinge} we set $s(h) = -yh_{\text{hinge}}\eta$. In conclusion

$$s(h) = -y \min(h, h_{\text{hinge}})\eta$$

This matches the intuition when one thinks about the limit of infinitely many infinitely small updates: For large importance weights, the process will bring the prediction up to y and make no further progress.

The quantile loss is similar and the update rule first computes the importance weight h' that would take the updated prediction at the point of nondifferentiability and then multiplies the gradient by $\min(h, h')$.

3.2 Variable Learning Rate

To handle a decaying learning rate η_t, we just need to slightly modify (7). Let $\eta_t(u)$ be the value of the learning rate u timesteps after time t. Then (7) becomes

$$s'(h) = \eta_t(h) \left. \frac{\partial \ell}{\partial p} \right|_{p = (w_t - s(h)x_t)^\top x_t}, \quad s(0) = 0$$

The solutions in this case are not qualitatively any different from the solutions of (7). We just need to replace the occurrences of $h\eta$ with $\int_0^h \eta_t(u)du$. Again for popular choices of learning rate such as $\eta_t(u) = \frac{1}{(t+u)^p}$ with $p = \frac{1}{2}$ or $p = 1$, this has a closed form.

3.3 Regularization

Theorem 1 can be modified to handle losses of the form $\ell(w^\top x, y) + \frac{\lambda}{2}||w||^2$ where $||w||$ is the Euclidean norm of w. However, the resulting differential equation is considerably harder and we have only been able to obtain a solution for the case of squared loss. Therefore we describe an alternative way of incorporating regularization based on a splitting approach [8]. First perform h unconstrained steps using the closed form solution, then compute the effect of regularization:

$$w_{t+1} = \operatorname*{argmin}_w \frac{1}{2}||w - (w_t - s(h)x)||^2 + \frac{h\eta\lambda}{2}||w||^2.$$

Note that we apply all h regularizers at once. The solution to the above optimization problem is

$$w_{t+1} = \frac{w_t - s(h)x_t}{1 + h\eta\lambda}$$

This approach can also handle other regularizers such as $\lambda||w||_1$ leading to a truncated gradient update [17].

4 PROPERTIES OF THE UPDATES

4.1 Invariance

First we show that the invariance property we mentioned in the introduction holds. It is convenient to explicitly state the dependence on the prediction p, by writing $s(p, h)$ instead of $s(h)$. The following theorem

Table 1: Importance Invariant and Imp^2 (cf. section 5) Updates for Various Loss Functions

Loss	$\ell(p,y)$	Invariant Update $s(h)$	Imp^2 Update
Squared	$\frac{1}{2}(y-p)^2$	$\frac{p-y}{x^\top x}\left(1-e^{-h\eta x^\top x}\right)$	$\frac{h\eta(p-y)}{1+h\eta x^\top x}$
Logistic	$\log(1+e^{-yp})$	$\frac{W(e^{h\eta x^\top x+yp+e^{yp}})-h\eta x^\top x-e^{yp}}{yx^\top x}$ for $y\in\{-1,1\}$	Not Closed
Exponential	e^{-yp}	$\frac{py-\log(h\eta x^\top x+e^{py})}{x^\top xy}$ for $y\in\{-1,1\}$	Not Closed

Logarithmic	$y\log\frac{y}{p}+(1-y)\log\frac{1-y}{1-p}$	if $y=0$ $\frac{p-1+\sqrt{(p-1)^2+2h\eta x^\top x}}{x^\top x}$
		 if $y=1$ $\frac{p-\sqrt{p^2+2h\eta x^\top x}}{x^\top x}$
			Not Closed

Hellinger	$2(1-\sqrt{py}-\sqrt{(1-p)(1-y)})$	if $y=0$ $\frac{p-1+\frac{1}{4}(12h\eta x^\top x+8(1-p)^{3/2})^{2/3}}{x^\top x}$
		 if $y=1$ $\frac{p-\frac{1}{4}(12h\eta x^\top x+8p^{3/2})^{2/3}}{x^\top x}$
			Not Closed

Hinge	$\max(0,1-yp)$	$-y\min\left(h\eta,\frac{1-yp}{x^\top x}\right)$ for $y\in\{-1,1\}$	Same

τ-Quantile	if $y>p$ $\tau(y-p)$
	 if $y\le p$ $(1-\tau)(p-y)$
		if $y>p$ $-\tau\min(h\eta,\frac{y-p}{\tau x^\top x})$
		 if $y\le p$ $(1-\tau)\min(h\eta,\frac{p-y}{(1-\tau)x^\top x})$
			Same

states that an update with an importance weight $a+b$ is equivalent to an update with importance a immediately followed by an update with importance b.

Theorem 2. *Let $s(p,h)$ be the solution of*

$$\frac{\partial s}{\partial h}=\eta\left.\frac{\partial \ell}{\partial p}\right|_{p=w^\top x-s(p,h)x^\top x}, \quad s(p,0)=0$$

where ℓ is a continuously differentiable loss. Then

$$s(p,a+b)=s(p,a)+s(p-s(p,a)x^\top x,b)$$

The proof is in the full version [13] and uses the Existence and Uniqueness Theorem for ODEs.

4.2 Safety

For some loss functions such as squared loss, hinge loss and quantile loss the residual $w_t^\top x_t-y_t$ tells us whether the learner is overestimating or underestimating the target. We call an update safe if

$$\frac{w_{t+1}^\top x_t-y_t}{w_t^\top x_t-y_t}\geq 0$$

whenever $(w_t^\top x_t-y_t)\neq 0$. Since the residual does not change sign after a safe update, it leads to sane results even when the learning rate is very aggressive.

Standard gradient descent is not safe, and importance invariant step sizes always are. This should be obvious for the hinge loss and the quantile loss because they use the minimum necessary step. For squared loss:

$$\begin{aligned}\frac{w_{t+1}^\top x-y}{w_t^\top x-y} &= \frac{w_t^\top x+\frac{y-w_t^\top x}{x^\top x}\left(1-e^{-h\eta x^\top x}\right)x^\top x-y}{w_t^\top x-y}\\ &= e^{-h\eta x^\top x}>0\end{aligned}$$

hence the update is safe.

4.3 Fallback Regret Analysis

Here we provide a fallback analysis for the case $h_t=1$. For simplicity we only show the results for squared loss and $||x_t||=1$ for all t. However, this can be extended to other losses as a Taylor expansion of each update around $\eta=0$ shows that to first order, it is equivalent to online gradient descent. Hence we expect a regret analysis similar to the one achieved by the underlying learning rate schedule. A proof of the following theorem is in the full version [13].

Theorem 3. *If $\ell(p,y)=(p-y)^2$ and $||x_t||=1$ for all t then the importance invariant update attains a regret of $O(\sqrt{T})$ when $\eta_t=\frac{1}{\sqrt{t}}$, and a regret of $O(\log(T))$ when $\eta_t=\frac{1}{t}$.*

5 IMPLICIT IMPORTANCE WEIGHT UPDATES

Implicit updates, first proposed in [14] and recently analyzed in [15] provide an alternative way for handling importance weights. An implicit update sets:

$$w_{t+1}=\operatorname{argmin}\frac{1}{2}||w-w_t||^2+\lambda\ell(w^\top x,y)$$

where λ is a free parameter similar to the learning rate in interpretation. Finding the minimizing w generally requires an iterative root-finding algorithm which is perhaps an order of magnitude more expensive than the closed form updates we derive above, although often easily amortized by the update itself. For squared loss and hinge loss closed-form solutions have been known. In the full version [13] we show how to derive these as well as a closed form implicit update for quantile loss which, to the best of our knowledge, is new. To adapt implicit updates for importance weights we

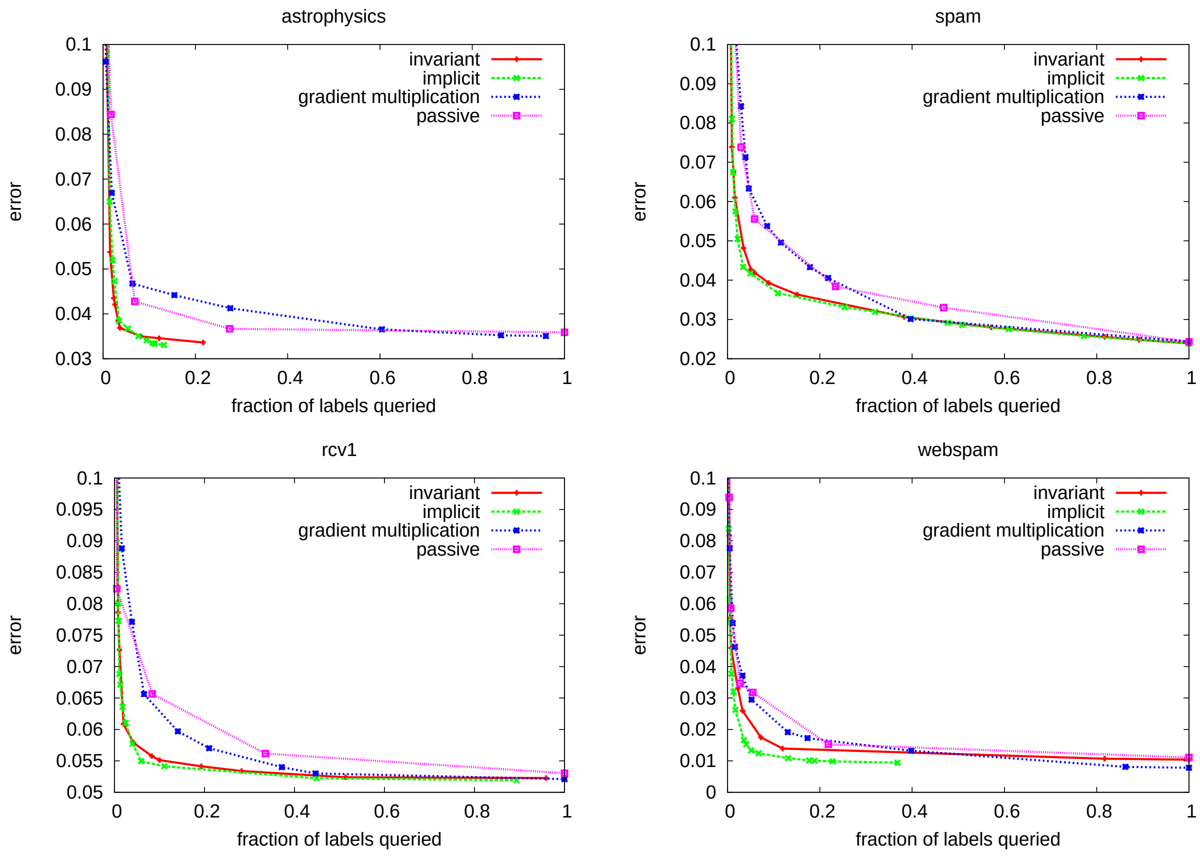

Figure 1: Test error vs. fraction of queried labels for each dataset

simply use $\lambda_t = \eta h_t$ (or $\lambda_t = \int_0^{h_t} \eta_t(u)du$ as in section 3.2) yielding an algorithm we call Imp^2. Imp^2 has qualitatively similar properties, satisfying Safety and Regret, but not Invariance or a Closed form update. In fact, Imp^2 for hinge and quantile is precisely equivalent to the importance invariant update.

6 EXPERIMENTS

We present empirical results on four text classification datasets: 'rcv1' is a modified version [16] of RCV1 [18], 'astro' is from [12], 'spam' was created from the TREC 2005 spam public corpora, and 'webspam' is from the PASCAL large scale learning challenge. In all experiments we did a single pass through the training set and we report the error on the test set. We try all learning rate schedules of the form $\eta_t = \frac{\mu}{x_t^\top x_t}\left(\frac{\tau}{t+\tau}\right)^p$ with $(\mu, \tau, p) \in \{2^i\}_{i=0}^{10} \times \{10^i\}_{i=0}^{8} \times \{0.5, 1\}$.

In the first set of experiments, large importance weights will inevitably appear. In particular, we treat all four datasets as active learning tasks and apply the algorithm in [2]. At each timestep t this algorithm computes a probability of querying the label of a point based on a quantity G_t that measures the difference in error rates between two hypotheses. The empirical risk minimizing (ERM) hypothesis and an alternative hypothesis that minimizes the empirical risk subject to predicting a different label than the ERM hypothesis. In our case the hypothesis we are learning with online gradient descent acts as the ERM hypothesis. To get a handle on G_t we estimate $t \cdot G_t$ by the importance weight that the example would need to have in order for an update with the alternative hypothesis's prefered label to cause the classification of the example to become the alternative label. In the full version [13] we derive the relevant importance weights for invariant and implicit updates. Once we have an estimate for G_t we can compute the query probability P_t and, if we query the label, we add the example to our dataset with importance weight $1/P_t$ to keep things unbiased. The query probabilities turn out to be proportional to the learning rate η_t and hence the algorithm will generate importance weights of order $O(\eta_t^{-1})$ growing at least as fast as $\Omega(\sqrt{t})$. Notice that importance weights are used for two different tasks: estimating G_t and preserving unbiasedness. Our linear model (with a link function $\sigma(p) = \max(0, \min(1, p))$) is optimizing squared loss. Details are in the full version [13].

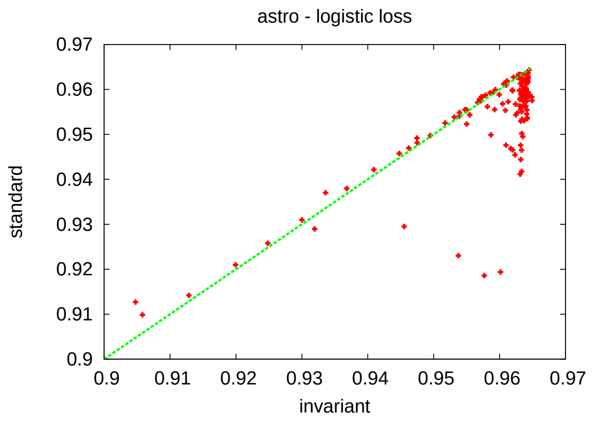

Figure 2: Scatter plot showing test accuracy with two different updates for various datasets and losses

In Figure 1 we summarize the results of the active learning experiments. Each combination of learning rate schedule and setting of the parameter C_0 in the active learning algorithm ($C_0 \in \{10^{-8}, 10^{-7}, \ldots, 10^1\}$) is an experiment that can be represented in the graph by a point whose x-coordinate is the fraction of labels queried by the active learning algorithm and whose y-coordinate is the test error of the learned hypothesis. To summarize this set of points, the figures plot part of its convex hull. The points on the convex hull (sometimes called a Pareto frontier) are experiments which represent optimal tradeoffs between generalization and label complexity, for some setting of this tradeoff. When a curve stops sooner than the size of the dataset it means that there were no experiments in which using more queries gave better generalization. We have also included the results from a typical good run of a passive learner. The graphs show very convincingly the value of having an update that handles importance weights correctly. Doing so yields better generalization and lower label complexity, than those attainable by multiplying the gradient with the importance weight. In fact, table 2 which lists the ratio of labels between passive and active learning to achieve a given accuracy, shows that *linearization can make active learning need more labels than passive learning.*

In the second set of experiments we used the same datasets but treated all examples as having an importance weight of one. We compare standard online gradient, vs. invariant and implicit updates on four loss functions: squared, logistic, hinge and quantile($\tau = 0.5$) loss. The purpose of these experiments is to highlight the robustness of the invariant updates: they yield good generalization with little search for a good learning rate schedule (also noted in [15] for implicit updates). However, we begin with table 3 which shows the test accuracy of the hypotheses learned by each update after *exhaustively searching* over the learning rate schedule. For astro and rcv1 the differences are very small. The spam dataset is not TF-IDF processed, and there we see a substantial improvement with either new update. The results on the webspam dataset were initially puzzling, but we have verified that this is not a failure to optimize well; on the contrary the proposed updates *attain smaller progressive validation loss* [3] than standard online gradient descent on the training data. Since progessive validation loss deviates like a test set, this is evidence that the webspam test set has a different distribution from the training set.[1]

Table 2: Reduction in label complexity

Dataset	astro	rcv1	spam	web
Desired Accuracy	0.963	0.943	0.967	0.986
Multiplication	0.45	1.59	1.28	0.54
Implicit	5.12	6.55	1.88	4.33
Invariant	7.56	6.55	2.13	1.82

Table 3: Test accuracies (grid search over schedules)

Dataset	Loss	Invariant	Imp^2	Standard
astro	hinge	0.96626	Same	0.96694
	logistic	0.96494	0.96485	0.96432
	quantile	0.96629	Same	0.96703
	squared	0.96463	0.96429	0.96469
rcv1	hinge	0.94872	Same	0.94838
	logistic	0.94704	0.94682	0.94743
	quantile	0.94846	Same	0.94859
	squared	0.94769	0.94799	0.94790
spam	hinge	0.97626	Same	0.97411
	logistic	0.96676	0.97982	0.97603
	quantile	0.97524	Same	0.97484
	squared	0.97609	0.97614	0.97563
web	hinge	0.98936	Same	0.99142
	logistic	0.99094	0.99038	0.9923
	quantile	0.98908	Same	0.99088
	squared	0.98960	0.98966	0.99218

To illustrate robustness we present the results in two ways. First, in Figure 2 we show a scatterplot where each point is a learning rate schedule and its coordinates are the accuracy of the learned hypothesis with and without the proposed updates. Scatter plots for other loss functions and datasets look very similar and are included in the full version [13]. The plot only shows the cases when both learning rates achieve accuracy above 0.9 and there are virtually no schedules for which one update is superior by more than 0.1. Among these cases the vast majority of experiments are clustered under the $y = x$ line and towards the

[1]The test set consists of the last 50000 examples of the original training set. The real test labels are not public.

Table 4: Fraction of schedules with near optimal error

Loss	Invariant	Standard
hinge	0.337	0.039
logistic	0.109	0.050
quantile	0.361	0.053
squared	0.306	0.031

extreme values of the x-axis. Consequently, when using the importance invariant update many schedules provide excellent performance.

To make this more clear we present a second way of viewing this result. In table 4 we report the fraction of learning rate schedules that achieve generalization accuracy within 0.001 of the best learning rate schedule, on average across all four datasets. For loss functions for which there is a notion of overshooting and can benefit from a safe update we observe an order of magnitude improvement in the number of schedules that converge to near optimal performance.

7 CONCLUSIONS

We tuned online gradient descent learning algorithms for various losses so they efficiently incorporate importance weight information, as is needed for applications in boosting, active learning, transfer learning, and learning reductions. The essential lesson here is that taking into account the curvature of the loss function can be done cheaply and provides great benefits in dealing with importance weights.

Motivated by an invariance property we proposed new updates that improve the standard update rule even for the baseline importance weight 1 case, yielding better prediction performance while simultaneously reducing the value of learning rate parameter search. Experiments not reported here show that it even improves the performance of adaptive gradient descent methods such as [4, 7]. Since this tuned update rule is computationally "free" we expect wide use.

Acknowledgements

We thank Robert Kleinberg for help with the proof of Theorem 2 and Alex Strehl for insightful discussions.

References

[1] A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *ICML*, 2009.

[2] A. Beygelzimer, D. Hsu, J. Langford, and T. Zhang. Agnostic Active Learning Without Constraints. *NIPS*, 2010.

[3] A. Blum, A. Kalai, and J. Langford. Beating the holdout: Bounds for kfold and progressive cross-validation. In *COLT*, 1999.

[4] H. Brendan McMahan and M. Streeter. Adaptive Bound Optimization for Online Convex Optimization. 2010.

[5] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

[6] O. Dekel, S. Shalev-Shwartz, and Y. Singer. Smooth ε-Insensitive Regression by Loss Symmetrization. *Journal of Machine Learning Research*, 6:711–741, 2005.

[7] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. 2010.

[8] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934, 2009.

[9] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*. Springer, 1995.

[10] M. Herbster. Learning additive models online with fast evaluating kernels. In *Computational Learning Theory*. Springer, 2001.

[11] J. Huang, A.J. Smola, A. Gretton, K.M. Borgwardt, and B. Scholkopf. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19, 2007.

[12] T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006.

[13] Nikos Karampatziakis and John Langford. Importance weight aware gradient updates. *CoRR*, abs/1011.1576, 2010.

[14] J. Kivinen and M.K. Warmuth. Exponentiated Gradient versus Gradient Descent for Linear Predictors. *Informatics and Computation*, 132:1–64, 1997.

[15] B. Kulis and P. Bartlett. Implicit Online Learning. In *ICML 2010*, 2010.

[16] J. Langford, L. Li, and A. Strehl. Vowpal wabbit online learning project, 2007. `http://hunch.net/?p=309`.

[17] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *The Journal of Machine Learning Research*, 10:777–801, 2009.

[18] D.D. Lewis, Y. Yang, T.G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[19] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[20] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *ICDM*. IEEE, 2003.