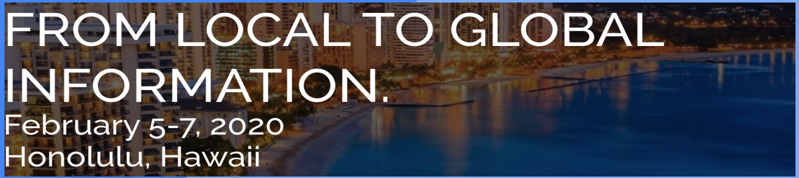
Leveraging Local Information in Practical Machine Learning:

Minimum Description Length Regularization for Online Learning

Gil I. Shamir

Presented, Feb 6, 2020 at:



Overview

Problem and Approach:

- Online learning
- Online feature selection - *locally per feature*
- Minimum Description Length (MDL) principle applied

Main Results:

- Improved sparsity accuracy tradeoffs
- Mitigation of overfitting

Outline

- The Problem
- Stochastic Gradient Descent
- Sparsity and regularization
- Standard sparsity regularization methods
- Minimum Description Length (MDL)
- MDL for online regularization
- Empirical results
- Analysis

General Learning Problem Setting

- A model $\mathcal{M}$ has:
  - Large set of features
  - Model features have (unknown) weights **x**

General Learning Problem Setting

- A model $\mathcal{M}$ has:
  - Large set of features
  - Model features have (unknown) weights **x**
- Learn the weights **x** from training examples (**a**, y)
  - **a** - sparse vector of feature values
  - y - label of example

General Learning Problem Setting

- A model $\mathcal{M}$ has:
  - Large set of features
  - Model features have (unknown) weights **x**
- Learn the weights **x** from training examples (**a**, y)
  - **a** - sparse vector of feature values
  - y - label of example
- Goal - Minimize some loss
  - Over all training examples
  - **For unseen examples**

General Learning Problem Setting

- A model $\mathcal{M}$ has:
  - **Very** large set of features
  - Model features have (unknown) weights **x**
- Learn the weights **x** from training examples (**a**, y)
  - **a** - sparse vector of feature values
  - y - label of example
- Goal - Minimize some loss
  - Over all training examples
  - **For unseen examples**

General Learning Problem Setting

- A model $\mathcal{M}$ has:
  - **Very very** large set of features (Google scale)
  - Model features have (unknown) weights **x**
- Learn the weights **x** from training examples (**a**, y)
  - **a** - sparse vector of feature values
  - y - label of example
- Goal - Minimize some loss
  - Over all training examples
  - **For unseen examples**

Google™

Online Learning Problem Setting

- At round t:
  - We have
    - Estimate of $\mathbf{x}_t$
  - We know
    - Features values $\mathbf{a}_t$
  - We try to estimate
    - Label y_t

Estimates based on past t-1 rounds.

Examples of Practical Problems

- Click-Through-Rate (CTR) prediction for ads
  - Feature examples:
    - Words in query
    - Words in ads
  - Label: clicked/not-clicked
  - Prediction: probability of a click
- Classification of products to categories
  - Label: which category
  - Prediction: probability of each category

Example: Logistic Regression with Log Loss

- Feature vectors:
  - Values at round: $\mathbf{a}_t \in \{0, 1\}$ or other range.
  - Prediction weights (log-odds): $\mathbf{x}_t$
  - Label $y_t \in \{0, 1\}$
- Predicted positive (1) label probability:

$$p_t = \sigma(\mathbf{a}_t \cdot \mathbf{x}_t) \qquad \sigma(z) \triangleq \frac{1}{(1 + e^{-z})}$$

- Loss:

$$f_t(\mathbf{x}) = -y_t \log p_t - (1 - y_t) \log(1 - p_t)$$

Formal Online Convex Optimization Setting

Online Convex Optimization:

- Series of rounds $t \in \{1, 2, \ldots, T\}$
- Play $\mathbf{x}_t \in \mathbb{R}^k$ at round t
- Incur loss $f_t(\mathbf{x}_t)$ at t
- Try to minimize overall loss

Regret: relative to fixed comparator $\mathbf{x}^*$

$$\text{Regret}(\mathbf{x}^*) \triangleq \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^*)$$

Google

General Soluation - SGD

Stochastic Gradient Descent (SGD)

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{g}_t$$

- Loss gradient: $\mathbf{g}_t = \nabla f_t(\mathbf{x}_t)$
- Step size: η_t
  - Per round
  - Per coordinate [Duchi *et. al*., Streeter/McMahan, 2010-11]

Google

Sparsity

The problems:

- **Some features are useless for prediction**
  - Inject noise (increasing regret)
  - Increase model size
- **We can't deploy models so big**

Regularization - Standard Approach

What is regularization?

- **Introduce additional loss component**
  - Constrains weights
  - Adds mathematical convenience
  - Can be viewed as *prior* on weights

L1 Regularization - Batch Approach

L1 Regularization

- **Additional L1 Norm based loss**

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ \sum_{t=1}^{T} f_t(\mathbf{x}) + \lambda_1 \|\mathbf{x}\|_1 \right\}$$

  - Forces weights to 0.
  - Mathematically convenient
  - Laplace *prior*

L1 - Online Shrinkage

Continuously shrink weights toward 0 when updated

[Beck & Teboulle, 09]

- Motivated from batch derivation
- Shrinks noisy features
  - But also good ones
- Very sensitive to shrinkage parameter
- No direct derivation for online

Follow The Regularized Leader (FTRL)

- A general approach
- SGD - special case

Can be used to formulate online L1 regularization

- Mathematically provable

Follow The Regularized Leader (FTRL)

- With strongly convex **regularizer** $r_t(\mathbf{x})$ play

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}} \{f_{1:t}(\mathbf{x}) + r_{0:t}(\mathbf{x})\}$$

$$f_{1:t} = \sum_t f_t \qquad r_{0:t} = \sum_t r_t$$

- Linearization: replace loss by: $\bar{f}_t(\mathbf{x}) = \mathbf{g}_t \cdot \mathbf{x}_t$
  - [Zinkevich, 03]
- Linearized version becomes SGD with
  - [McMahan, 11]

$$r_t(\mathbf{x}) = \frac{\varphi_t}{2}\|\mathbf{x} - \mathbf{x}_t\|_2^2, \quad \eta_t \triangleq \frac{1}{\varphi_{1:t}}$$

L1 with FTRL

- Add L1 term to FTRL
  - [McMahan, 11]

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}} \left\{ \mathbf{g}_{1:t} \cdot \mathbf{x} + r_{0:t}(\mathbf{x}) + \lambda_1 \left\| \mathbf{x} \right\|_1 \right\}$$

Still suboptimal Regularization!

Drawbacks

Performance

- Suboptimal accuracy sparsity tradeoffs
- **Underfitting** or **overfitting**

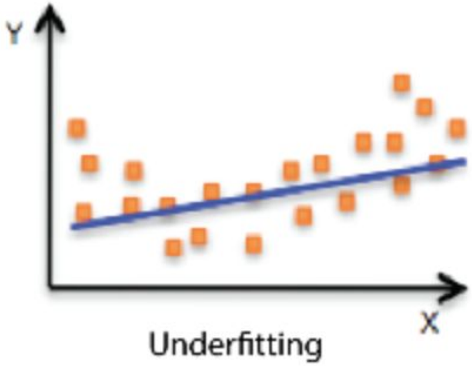
Why?

- Maximum likelihood (ML) ignores cost of parameters
- L1 methods select (Laplace) prior on weights
  - Does not rely directly on objective
    - Shrinkage - does not let good weights converge
    - Fixed threshold - lets noise in

Underfitting

What is underfitting?

- Poor predictions
  - Model does not capture all data
  - **Missing features**

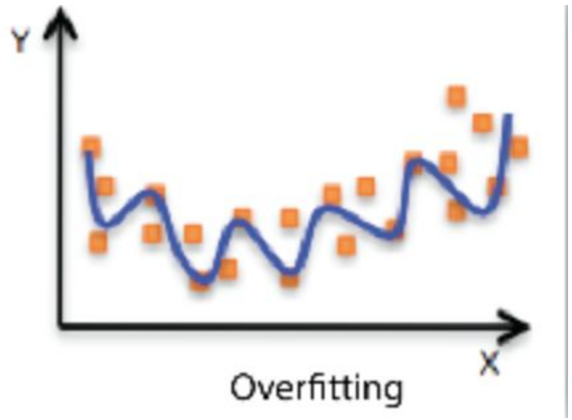


[From AWS ML]

Overfitting

What is overfitting?

- Poor predictions
  - Model captures noise with data
  - **“Too many features”**
    - Including useless / harmful ones.
- Predict well on training data
  - Poorly on unseen data



[From AWS ML]

Standard Methods - [Rissanen, 89]

“We never want to make the **false** assumption that the observed data actually were generated by a distribution of some kind, say Gaussian, and then go on to analyze the consequences and make further deductions. Our deductions may be **entertaining** but quite **irrelevant** to the task at hand, namely, to learn useful properties from the data.”

Minimum Description Length (MDL)

The Principle:

- To describe data $\mathbf{y}_1^T$ choose the model $\mathcal{M}$ that minimizes

$$\mathcal{L}(\mathbf{y}_1^T) = \underbrace{\mathcal{L}(\mathcal{M})}_{\text{Model Cost}} + \underbrace{\mathcal{L}(\mathbf{y}_1^T|\mathcal{M})}_{\text{Data Description Cost}}$$

[Rissanen’78, 84, 86]

MDL - Feature (Local) Level

- **Each Feature**
  - **Value/benefit -** potential improvement to loss
  - **Cost - learning of weight reflected in loss**
- **We must:**
  - Not ignore cost
  - Treat the sum of value and cost as **overall benefit** of feature

⟹ Model only includes features with overall positive benefit!

Advantages of MDL

Address problems with Standard L1 Methods

Overfitting Interpretation:

Features cost more to learn than benefit they bring

$\Longrightarrow$ No predictive value to unseen data

Underfitting:

- Include all features
  - With overall positive benefit

MDL Motivation - [Grunwald, 04]

“MDL procedures automatically and inherently protect against overfitting and can be used to estimate both the parameters and the structure (e.g., number of parameters) of a model. In contrast, to avoid overfitting when estimating the structure of a model, traditional methods such as maximum likelihood must be modified and extended with additional, typically *ad hoc* principles.”

So Where is MDL Used - Examples

- Universal data compression
  - Context Tree Weighting (CTW) [Willems, Shtarkov, Tjalkens 95]

- Offline model selection and denoising
  - For regression [Hansen, Yu 01]
  - Denoising [Rissanen, 00; Roos, Rissanen, 09]
  - Many more examples

MDL Offline Feature Selection

[Hansen, Yu 01]

- Select features prior to training by
  - Gain on loss
  - Lower bound trainings cost (0.5 log n per parameter)
- Mix multiple models
  - Mixture is function of lower bounds on loss
- Drawbacks
  - Infeasible
  - Suboptimal

Why MDL Wasn't Used

- Advantages Understood
  - [Rissanen, Grunwald]
  - [Zhao, Yu 06]
    - "L1 methods may not select model correctly"
    - "MDL methods are always consistent"
- How to do it was not:
  - "Approach not feasible in practice" [Grunwald 04]
  - "MDL computationally intractable" [Zhao, Yu 06]

MDL Regularization

So how do we do it online?

For each feature:

- Train model with it
- Train model without it
- Loss improvement with feature - **benefit score**

Only use features with positive score for prediction

MDL Regularization - Model Level

- Update/train all features
  - good benefit
  - But also bad
- Features can move between categories
- **Predict only with features with sufficient benefit**
- Can wrap over any algorithm
- No additional complexity
- Ranks features by importance
- Tweaks to weight learned weight as function of benefit

Why does it work?

Difference in loss with any wrapped algorithm

- **Already captures MDL benefit**
- Loss already includes
  - Gain of adding feature
  - Cost of learning feature
    - with the wrapped algorithm

No need to work hard - already have all we need

Trading Sparsity / Accuracy

Better tradeoffs than L1

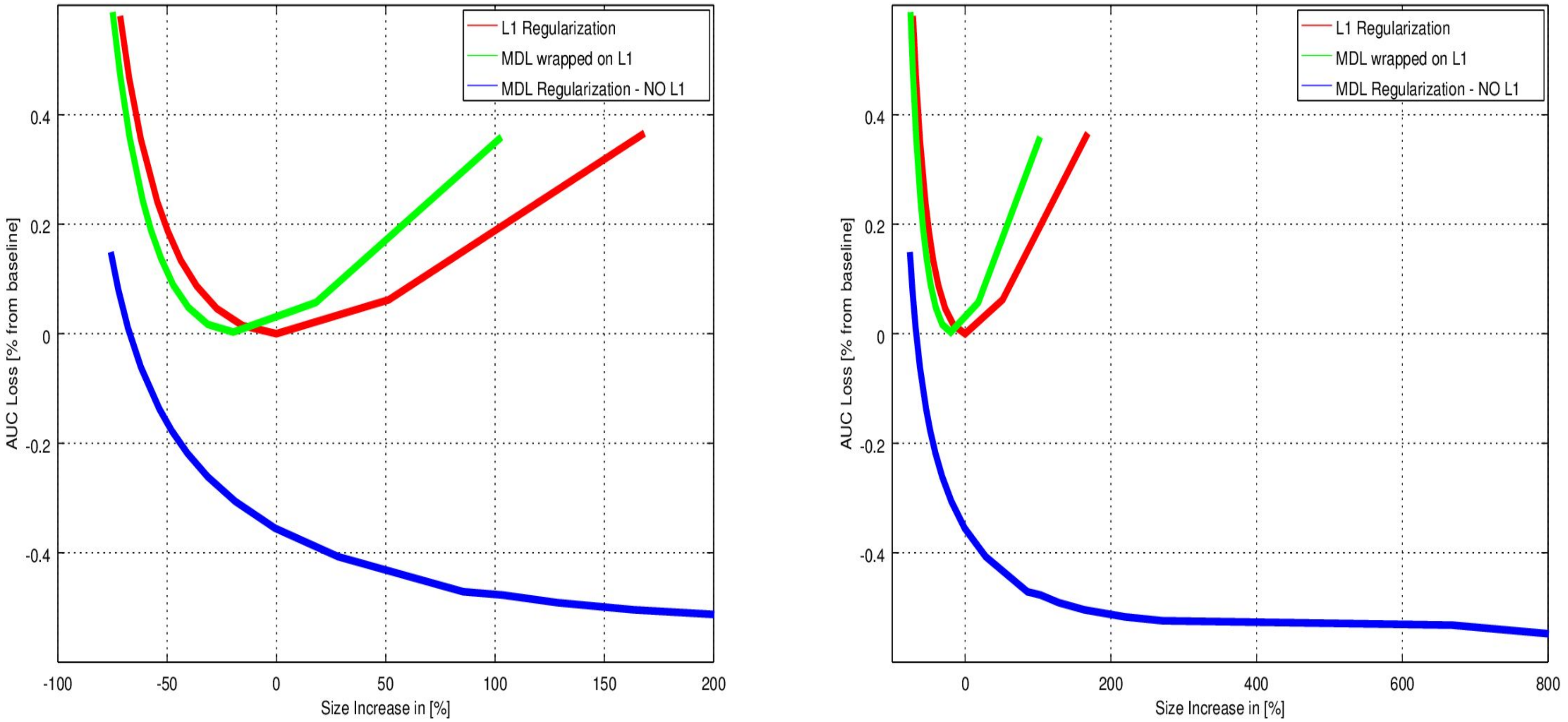
- Benefit based feature selection:
  - Benefit threshold μ
  - Benefit < $\mu \Rightarrow 0$ weight
- No overfitting when $\mu \geq 0$

Threshold	Model	Accuracy
Low	large	+
High	small	-

Experiments

- Click-Through-Rate (CTR) prediction
  - Huge set of features
  - Many many examples
- Several Algorithms
  - L1 FTRL (only)
  - MDL wrapped on L1 FTRL,
  - MDL on FTRL (no L1)
- Progressive validation AUC (Area under curve) loss
- Percent size and loss increase (decrease if negative)

Experimental Results



Results

- Better tradeoffs than L1
  - huge size reductions (~50%)
  - Better accuracy
- **No overfitting**
  - (overfitting is present with L1)
- MDL with L1 - only size reduction

How to Analyze

Present MDL regularization as Bayesian Mixture

1. All possible feature subspaces
2. Negligible loss - doing it per feature
3. Bounded loss - sparsifying

Complete MDL Mixture Heuristic Approach

For Logistic Regression:

- Predictor for every feature subspace **s** - total 2^k
- Mix with prior for feature usefulness α

$$w^{(\mathbf{s})} = \alpha^{||\mathbf{s}||_0}(1-\alpha)^{k-||\mathbf{s}||_0}$$

- Label sequence probability per (subspace) state:

$$\prod_t p_{\mathbf{s},t}^{y_t} \cdot (1 - p_{\mathbf{s},t})^{1-y_t}$$

Loss of best subspace+O(||s||) (negligible) regret achieved

Not really feasible for a huge number of features

Per Feature (Local) Mixture

General Idea

- Bayesian mixture per feature **with a 0 weight**.
- Mixture weighted using **benefit score**:
- Features train:
  - with regular **base** algorithm (SGD, FTRL, etc.)
  - Seeing other mixed features

Google

Linearized MDL Mixture Algorithm

Notation:

- $\bar{\mathbf{x}}_t$ - vector played by base algorithm
- $\tilde{\mathbf{x}}_t$ - feature level mixed vector
- $\mathbf{B}_t$ - benefit score vector
- $\bar{\mathbf{g}}_t$ - base gradient (other features mixed)

- $\mathbf{x}_{-i,t}$ - $\mathbf{x}_t$ with component i set to 0
- $\mathbf{x}_{+i,t}$ - $\mathbf{x}_t$ with component i set to $\bar{x}_{i,t}$

Linearized MDL Mixture Updates

- Gradient update:

$$\bar{\mathbf{g}}_t^{(i)} \in \partial f_t\left(\tilde{\mathbf{x}}_{+i,t}\right), \forall i$$

- Benefit update:

$$B_{i,t+1} \leftarrow B_{i,t} - \left[f_t\left(\tilde{\mathbf{x}}_{+i,t}\right) - f_t\left(\tilde{\mathbf{x}}_{-i,t}\right)\right], \forall i$$

- Base update:

$$\bar{\mathbf{x}}_{t+1} \leftarrow \arg\min_{\bar{\mathbf{x}}} \left[\bar{\mathbf{g}}_{1:t} \cdot \bar{\mathbf{x}} + r_{0:t}\left(\bar{\mathbf{x}}\right)\right]$$

- Mixture update:

$$\tilde{x}_{i,t+1} \leftarrow \sigma\left(B_{i,t+1} + \rho(\alpha)\right) \cdot \bar{x}_{i,t+1},$$

  - $\rho(\alpha)$ - function of prior
  - $\sigma(\cdot)$ - Sigmoid

Google™

How Does it Work?

- Benefit score
  - **Large** for **useful** features
  - **Small** for **useless** features
- Mixture weight
  - Approaches **1** quickly for **good** features
  - Approaches **0** quickly for **bad** features
- Base learning algorithm already does MDL
  - Benefit increases with “entropy” gain
  - Base algorithm cost incurred as algorithm learns
- 0 mass suppresses bad features **avoiding their MDL loss**

Linearized MDL Mixture Performance

Theorem:

- Under mild regularity conditions on the loss,
- The loss of the complete MDL mixture is achieved
  - (with additional terms negligible relative to the regret).

Explanation:

- Bound on Loss of best feature subspace achieved
  - (sum of “entropy” and regret terms)
- No complexity penalty

Balancing Accuracy and Sparsity

Problem: Mixture still uses all features

- Saves on accuracy
- Does not impose sparsity

Solution (already shown): Threshold benefit score per feature

- Below threshold - use as 0 for prediction
- Above threshold - use mixture weight

⇨ **MDL Regularization**

MDL Regularization Performance

Corollary:

MDL Regularization achieves the MDL mixture loss with an additional $O(\mu m)$ term

- m - useful features in best feature subspace
- μ - benefit threshold selected

Google

Summary and Conclusions

- Novel MDL based regularization
  - based on actual objective
- Better accuracy - sparsity tradeoffs
- Overfitting mitigated
- MDL mixture analysis leads to theoretical guarantees

Information Theory ideas $\Longrightarrow$ Improved practical systems

Local optimization $\Longrightarrow$ Global solution.