6.867 Fall 2017
Introduction to Classification

Logistic Regression

Lecture 6: 26th Sept., 2017

Admin

No recitation this Friday (student holiday)

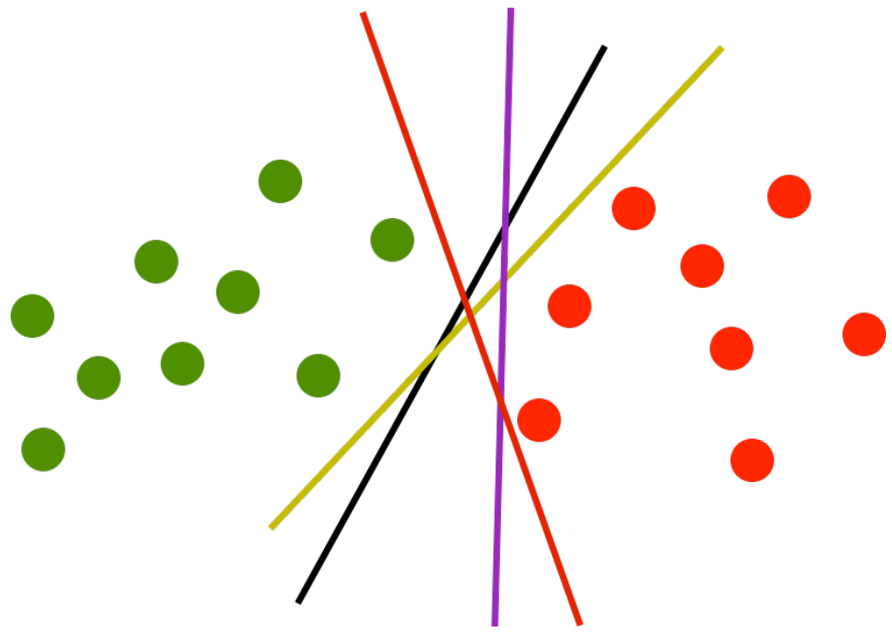
Project Milestone 0 (google form; link via stellar)

HW1 submission — CMT is thy friend

Projects: One writeup per team

Classification: recap

- first turn raw data into features
- we saw a linear classifier: the perceptron
- “mistake driven”: make mistake, shift hyperplane, repeat
- mistake bound for linearly separable case
- loss function formulation, perceptron via hinge-loss+SGD



Classification

Generative versus discriminative

Generative: Learn p(X,Y) using P(X|Y) and P(Y).
Use Bayes Rule to predict P(Y|X)

Discriminative: Directly learn the map P(Y|X)

Simpler (not solving a hard intermediate problem of getting P(X|Y)
Typically, computationally easier too
Often works better

NIPS 2002

On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes

Andrew Y. Ng
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720

Michael I. Jordan
C.S. Div. & Dept. of Stat.
University of California, Berkeley
Berkeley, CA 94720

Abstract

We compare discriminative and generative learning as typified by logistic regression and naive Bayes. We show, contrary to a widely-held belief that discriminative classifiers are almost always to be preferred, that there can often be two distinct regimes of performance as the training set size is increased, one in which each algorithm does better. This stems from the observation—which is borne out in repeated experiments—that while discriminative learning has lower asymptotic error, a generative classifier may also approach its (higher) asymptotic error much faster.

Logistic regression

Idea: Model the probability that the target Y belongs to a particular category

Assume that Y takes values in {0,1} (same idea, with ε care applies to {-1,1})

Question: How to model the relation between P(y=1|x) and x?

In linear regression we used:

$$\mu(x) = w_1 x + w_0$$

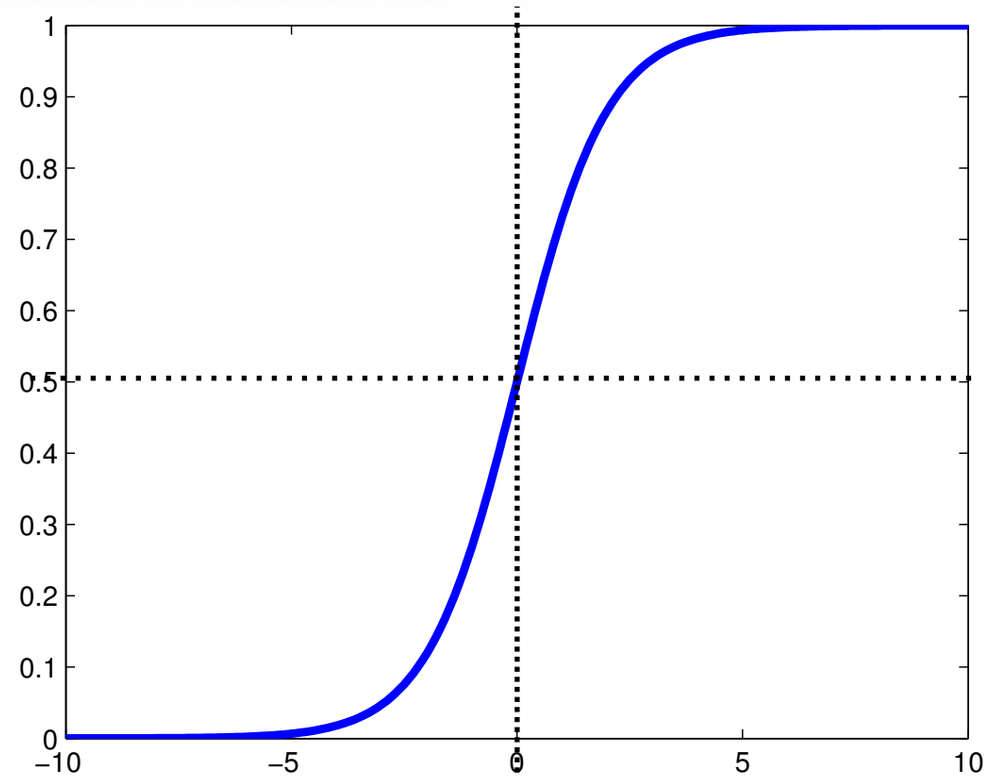
Not suitable here, we want μ to be a value in [0,1]

$$p(y|x, w) = \text{Bernoulli}(y|\mu(x))$$

$$\mu(x) = \mathbb{E}[y|x] = P(y = 1|x)$$

Logistic regression

$$\sigma(z) := \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$$



Sigmoid / logistic function

$$\mu(x) = \sigma(w_1 x + w_0) = \frac{e^{w_1 x + w_0}}{1 + e^{w_1 x + w_0}}$$

Observe how the sigmoid maps all of **R** into the interval [0,1]

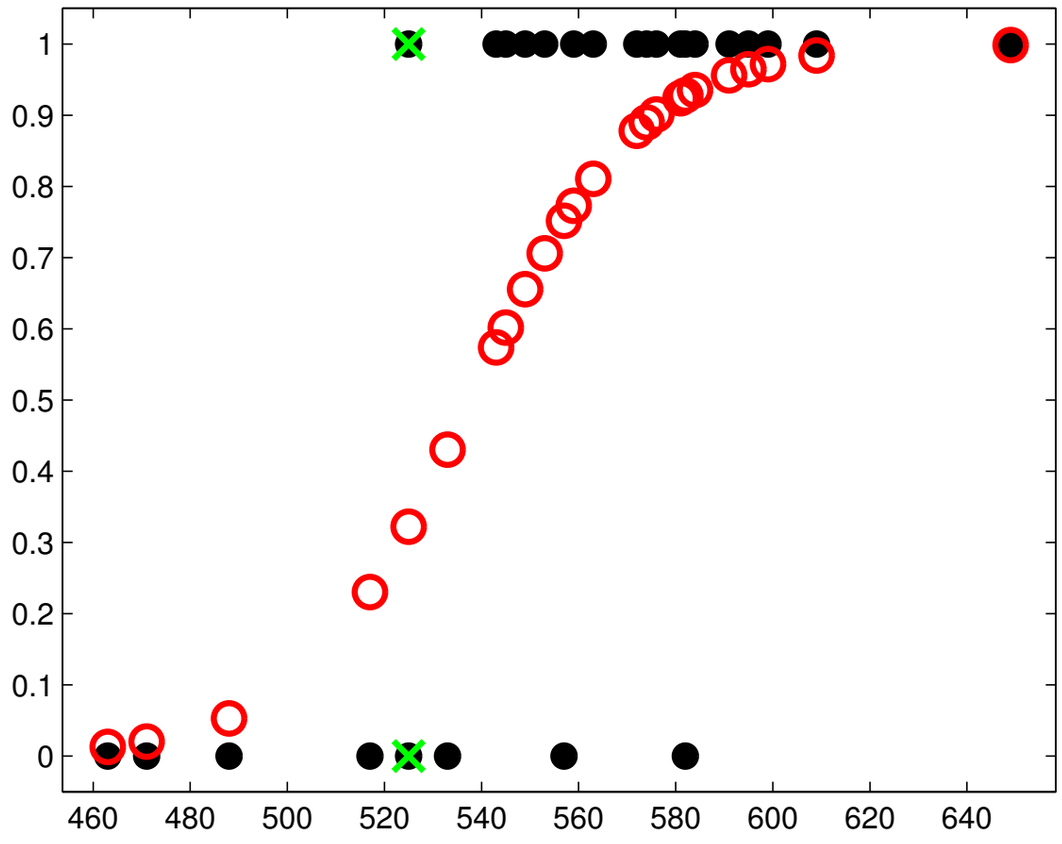
[figure credit: Kevin Murphy]

Logistic regression model

$$p(y|x,w) = \mathrm{Ber}(y|\sigma(w_1 x + w_0))$$

We plot (x_i is the SAT score)

$$p(y_i = 1|x_i, w) = \sigma(w_1 x_i + w_0)$$



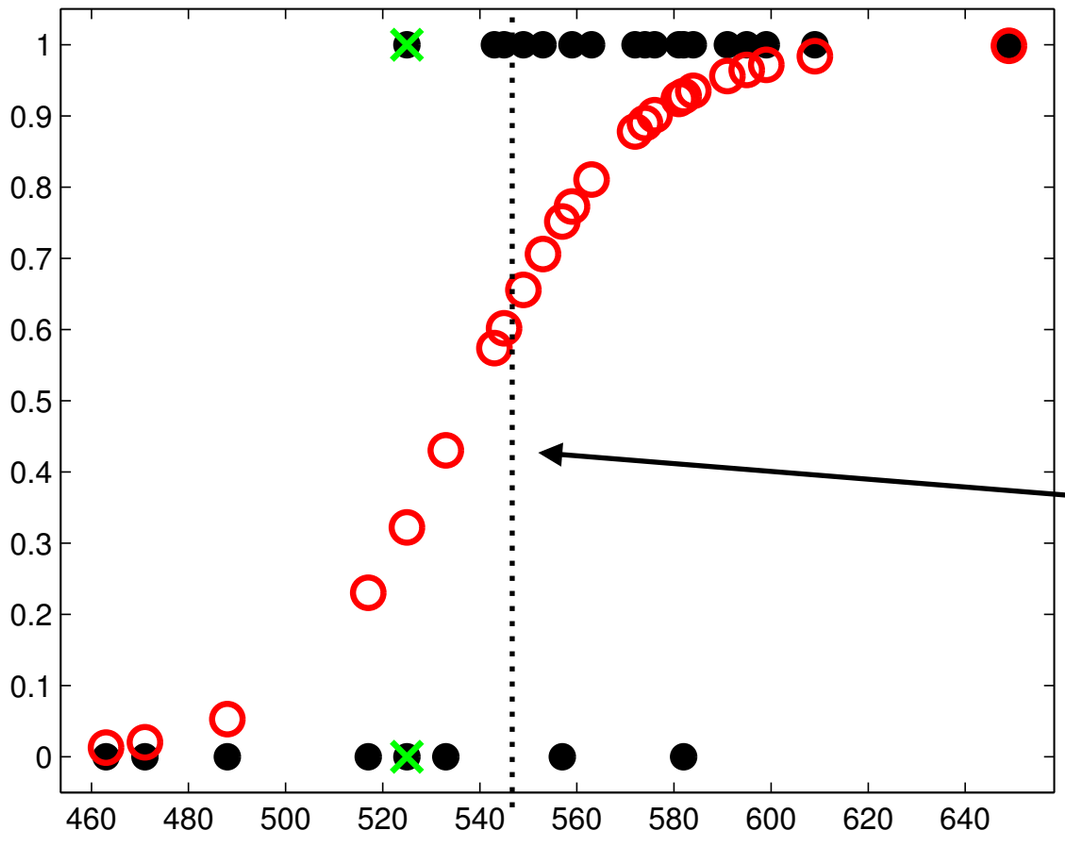
SAT score	***Pass / Fail ML101***
460	Fail
525	Pass
525	Fail
640	Pass
…	…

solid black: data;
red circles: predicted prob using LR

Logistic regression model

We plot (x_i is the SAT score)

$$p(y_i = 1|x_i, w) = \sigma(w_1 x_i + w_0)$$



solid black: data;
red circles: predicted prob using LR

Induce **decision rule**

$$\hat{y}(x) = 1 \quad \text{if} \quad p(y = 1|x) > 0.5$$

linear decision boundary

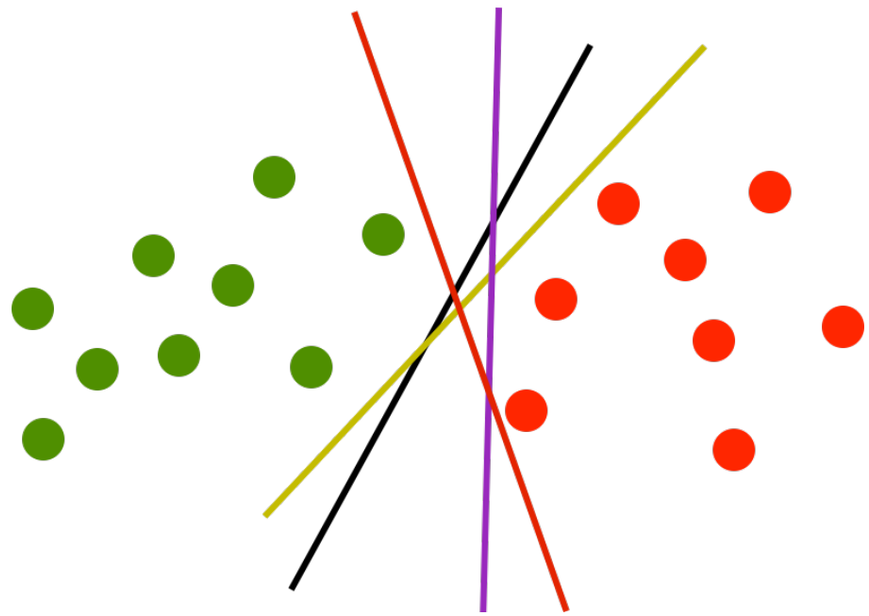
Observe: nonzero training error. Why?

LR decision boundary

$$\frac{1}{1+e^{-w^T x}} = 0.5$$

$$\implies e^{-w^T x} = 1$$

$$\implies w^T x = 0.$$



Something special happens to LR if data are linearly separable!

LR: some features

Easy to fit. In fact ideal example for even industrial scale data (we can build highly scalable parameter estimation methods)

Easy to interpret. Let us define the *log odds* as follows:

$$\mathcal{LO}(x) := \log \frac{p(y=1|x)}{p(y=0|x)} = w^T x$$

Example: Suppose two features: number of hours you study and time in days it took you to do homework. The goal is to predict the probability you will get an A.

Suppose, LR estimates the params to be w=(1.3,-1.1). This means that for every extra hour you study, your chance of getting an A increases by a factor of $e^{1.3}$

Easy to extend to multiclass (softmax), nonlinear features, kernels, or NNs

Some questions worth thinking about

- Why sigmoid and not something else?
- Regularization and Logistic Regression?
- What happens when the dataset is perfectly separable?
- How come logistic regression is a linear classifier?
- Interpreting coefficients of LR model

- Why LR and not LC?
- How well did LR explain my data? Was it a good choice?

LR: Maximum Likelihood Estimation

Aim: Given i.i.d. training data $\{(x_1,y_1),\ldots,\{x_N,y_N\})$, estimate param vector 'w'

Likelihood

$$\ell(w) = \prod_{i:y_i=1} p(x_i) \prod_{j:y_j=0} (1 - p(x_j))$$

$$p(x) \equiv \mu(x)$$

To maximize $\ell(w)$, we equivalently minimize negative log-likelihood.

$$\mathcal{L}(w) = -\sum_{i=1}^{N} \log \left[\sigma(w^T x_i)^{y_i} (1 - \sigma(w^T x_i))^{1-y_i}\right]$$

$$= -\sum_{i=1}^{N} \left[y_i \log \sigma(w^T x_i) + (1 - y_i) \log \left(1 - \sigma(w^T x_i)\right)\right]$$

LR: Maximum Likelihood Estimation

$$= -\sum_{i=1}^{N} \left[y_i \log \sigma(w^T x_i) + (1 - y_i) \log \left(1 - \sigma(w^T x_i)\right) \right]$$

This is called the **cross-entropy** error function

If we are using labels {-1,1}, then we have

$$p(y = 1|x) = \frac{1}{1 - e^{-w^T x}}, \quad p(y = -1|x) = \frac{1}{1 + e^{+w^T x}}$$

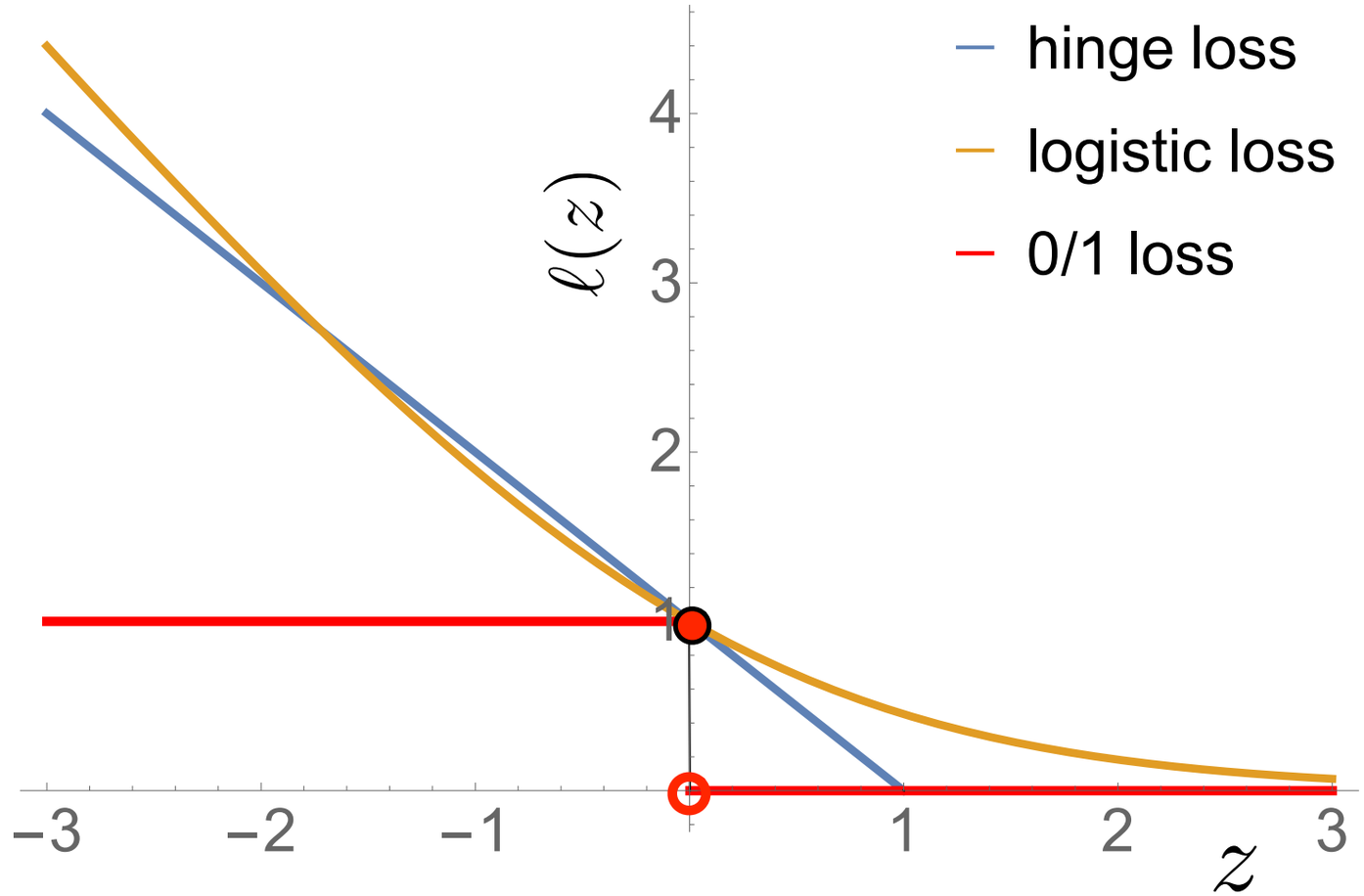
Logistic loss

$$\mathcal{L}(w) = \sum_{i=1}^{N} \log \left(1 + \exp(-y_i w^T x_i)\right)$$

Loss function viewpoint

Recall the **Empirical Risk Minimization** (ERM) problem

$$R_{\text{emp}}(w, w_0) := \sum_{i=1}^{N} \ell(y_i(w^T x_i + w_0))$$



- hinge loss $\ell_h(z) := \max(0, 1 - z)$
- logistic loss $\ell_{\log}(z) := \log(1 + e^{-z})$
- 0/1 loss $\ell_{0/1}(z) := [\![z \leq 0]\!]$

LR with logistic loss

$$\mathcal{L}(w) = \sum_{i=1}^{N} \log\left(1 + \exp(-y_i w^T x_i)\right)$$

Minimize using SGD (nice, differentiable loss function)

Algorithm:

1. Initialize weights. Set t = 0
2. For t=0,1,2,…
   - 2.1. Pick i in {1,…,N}
   - 2.2. Obtain subgradient of $\ell_{\log}(y_i w^T x)$
   - 2.3. Update $w^{t+1} = w^t - \eta_t g_t$

Stochastic convex optimization problem; SGD can be shown to converge to an optimum at the rate $O(1/\sqrt{T})$ (T updates).

Regularization for LR

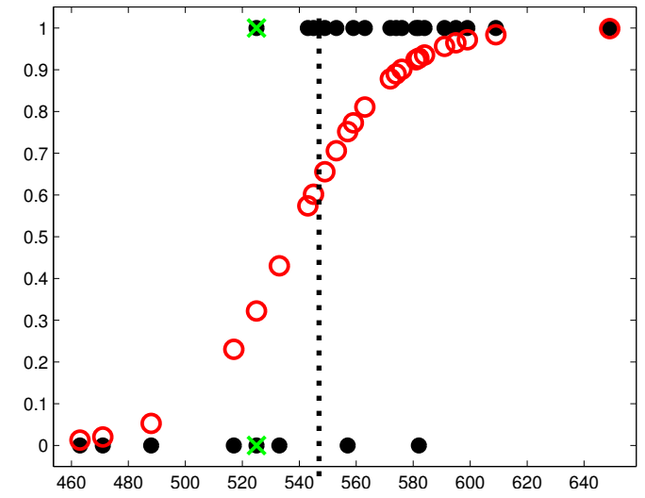
Suppose the data are linearly separable. In this case

$$w^T x_i > 0, \;\; y_i = 1$$

$$w^T x_i \leq 0, \;\; y_i = 0$$

$$\frac{e^{w^T x_i}}{1 + e^{w^T x_i}} \to \{1, 0\}, \quad \|w\| \to \infty$$

Model ends up assigning maximum amount of mass to the training data: **brittle / overfits**



Remark: MLE does not attain its minimum

Hint: $\sigma'(z) = \sigma(z)(1 - \sigma(z))$

Regularization for LR

Hence, as in ridge-regression, we add a control on ‘w’

$$\sum_{i=1}^{N} \log\left(1 + \exp(-y_i w^T x_i)\right) + \lambda \|w\|_q^q,$$

q=2: L2-norm regularized logistic regression

q=1: L1-norm regularized “sparse” logistic regression

Note: Adding an L2-regularizer makes the optimization problem much nicer (strongly convex, hence SGD will converge faster than without the regularizer)

Canonical convex optimization problem reported in ML papers that focus on large-scale convex optimization; actually has been widely used in industry too.

Large-scale logistic regression

Click Through Rate (CTR) prediction

Simple and scalable response prediction for display advertising

OLIVIER CHAPELLE, Criteo†
EREN MANAVOGLU, Microsoft
ROMER ROSALES, LinkedIn

Clickthrough and conversation rates estimation are two core predictions tasks in display advertising. We present in this paper a machine learning framework based on logistic regression that is specifically designed to tackle the specifics of display advertising. The resulting system has the following characteristics: it is easy to implement and deploy; it is highly scalable (we have trained it on terabytes of data); and it provides models with state-of-the-art accuracy.

http://people.csail.mit.edu/romer/papers/TISTRespPredAds.pdf

- Many ads sold on a “pay per click” basis (not “pay per view”)
- Aim of search engine, website, etc: choose which ad(s) to display based on *expected value* from an ad click.
- To maximize expected profit, maximize likelihood that given ad is clicked: this is the so-called *click through rate* (CTR)

CTR Prediction

Table I. Sample of features considered divided into feature families.

Feature family	Feature members
Advertiser	advertiser (id), advertiser network, campaign, creative, conversion id, ad group, ad size, creative type, offer type id (ad category)
Publisher	publisher (id), publisher network, site, section, url, page referrer
User (when avail.)	gender, age, region, network speed, accept cookies, geo
Time	serve time, click time

Very large number of categorical features: one-hot or one-hot-K encoding (`pandas get_dummies()`, `sklearn.preprocessing.OneHotEncoder`)

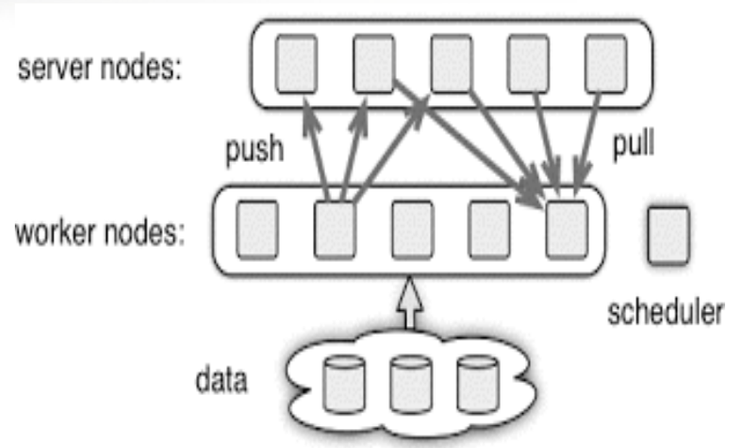
Leads to huge blowup in the number of features (i.e., dimensionality of the data matrix)

Number of samples is also very large, ranging in several billions of "impressions" per day

Aim: predict probabilities of click (y=1) or not click (y=-1)

Large-scale CTR prediction

$$\sum_{i=1}^{N} \log\left(1+\exp(-y_i w^T x_i)\right) + \lambda \|w\|_q^q,$$

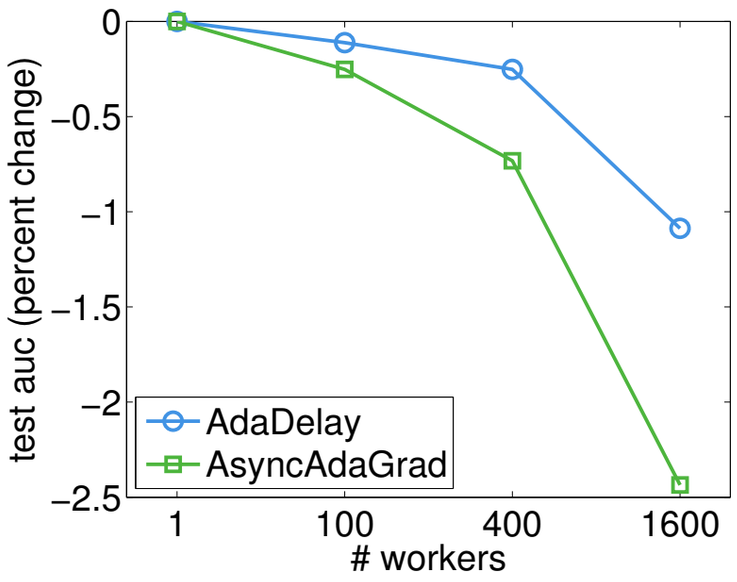


Distributed SGD on the cloud

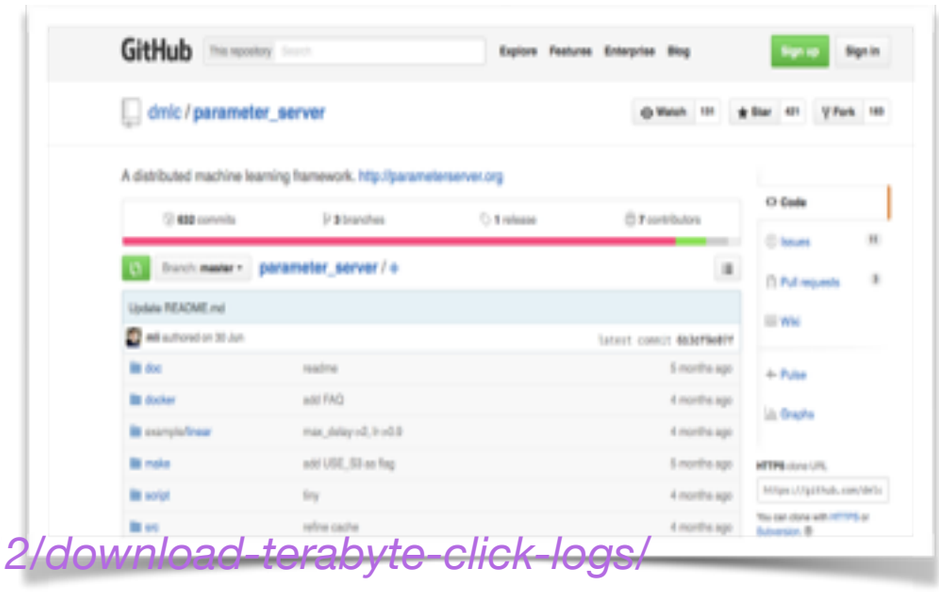
	training example	test example	unique feature	non-zero entry
Criteo	1.5 billion	400 million	360 million	58 billion
CTR2	110 million	20 million	1.9 billion	13 billion

Table 1: Click-through rate datasets.

AUC vs # worker threads



code at



http://labs.criteo.com/2013/12/download-terabyte-click-logs/