Slides adapted from Prof Carpuat and Duraiswami

CMSC 422 Introduction to Machine Learning

Lecture 14 Binary Classification with Linear Models

Furong Huang / furongh@umd.edu

UNIVERSITY OF MARYLAND

What we learned last time

- Ranking

- Bias and Fairness
  - Unsupervised adaptation

Unsupervised adaptation

Goal: learn a classifier f that achieves low expected loss under new distribution $\mathcal{D}^{new}$

Given labeled training data from old distribution

$\mathcal{D}^{\text{old}}$ $(x_1, y_1), \ldots, (x_N, y_N)$

And unlabeled examples from new distribution

$\mathcal{D}^{\text{new}}$: $z_1, \ldots, z_M$

Relation between test loss in new domain and old domain

$$
\begin{aligned}
&\text{test loss} && && (8.1)\\
&= \mathbb{E}_{(x,y)\sim\mathcal{D}^{\text{new}}}\left[\ell(y, f(\boldsymbol{x}))\right] && \text{definition} && (8.2)\\
&= \sum_{(x,y)} \mathcal{D}^{\text{new}}(\boldsymbol{x}, y)\ell(y, f(\boldsymbol{x})) && \text{expand expectation} && (8.3)\\
&= \sum_{(x,y)} \mathcal{D}^{\text{new}}(\boldsymbol{x}, y)\frac{\mathcal{D}^{\text{old}}(\boldsymbol{x}, y)}{\mathcal{D}^{\text{old}}(\boldsymbol{x}, y)}\ell(y, f(\boldsymbol{x})) && \text{times one} && (8.4)\\
&= \sum_{(x,y)} \mathcal{D}^{\text{old}}(\boldsymbol{x}, y)\frac{\mathcal{D}^{\text{new}}(\boldsymbol{x}, y)}{\mathcal{D}^{\text{old}}(\boldsymbol{x}, y)}\ell(y, f(\boldsymbol{x})) && \text{rearrange} && (8.5)\\
&= \mathbb{E}_{(x,y)\sim\mathcal{D}^{\text{old}}}\left[\frac{\mathcal{D}^{\text{new}}(\boldsymbol{x}, y)}{\mathcal{D}^{\text{old}}(\boldsymbol{x}, y)}\ell(y, f(\boldsymbol{x}))\right] && \text{definition} && (8.6)
\end{aligned}
$$

How can we estimate the ratio between Dnew and Dold?

Fixed base distribution

S = selection variable

$$\frac{\mathcal{D}^{\text{new}}(x,y)}{\mathcal{D}^{\text{old}}(x,y)} = \frac{\frac{1}{Z^{\text{new}}}\mathcal{D}^{\text{base}}(x,y)p(s=0 \mid x)}{\frac{1}{Z^{\text{old}}}\mathcal{D}^{\text{base}}(x,y)p(s=1 \mid x)} \quad \text{definition} \quad (8.9)$$

$$= \frac{\frac{1}{Z^{\text{new}}}p(s=0 \mid x)}{\frac{1}{Z^{\text{old}}}p(s=1 \mid x)} \quad \text{cancel base} \quad (8.10)$$

$$= Z\frac{p(s=0 \mid x)}{p(s=1 \mid x)} \quad \text{consolidate} \quad (8.11)$$

$$= Z\frac{1-p(s=1 \mid x)}{p(s=1 \mid x)} \quad \text{binary selection} \quad (8.12)$$

$$= Z\left[\frac{1}{p(s=1 \mid x)} - 1\right] \quad \text{rearrange} \quad (8.13)$$

We can estimate P(s=1|x) using a binary classifier!

Clarifications

- Note Z^{new} is the same for all the data points (i.e., $\forall(\boldsymbol{x}, y)$). It is also a normalization to make sure $\sum_{(x,y)} \mathcal{D}^{new}(x, y) = 1$.

$$Z^{new} = \sum_{(x,y)} [\mathcal{D}^{base}(x, y)p(s = 0|x)]$$

- Therefore, Z^{new} is not a function of $(\boldsymbol{x}, y)$, in fact it is a constant.
- Similarly for Z^{old}.
- All examples are drawn from fixed base distribution, some are selected to go into D^{new}, some are selected to go into D^{old} according to a selection variable s.

Algorithm 23 SELECTIONADAPTATION($\langle (x_n, y_n) \rangle_{n=1}^{N}$, $\langle z_m \rangle_{m=1}^{M}$, $\mathcal{A}$)

$D^{dist} \leftarrow \langle (x_n, +1) \rangle_{n=1}^{N} \bigcup \langle (z_m, -1) \rangle_{m=1}^{M}$ // assemble data for distinguishing // between old and new distributions

$\hat{p} \leftarrow$ train logistic regression on D^{dist}

$D^{weighted} \leftarrow \left\langle (x_n, y_n, \frac{1}{\hat{p}(x_n)} - 1) \right\rangle_{n=1}^{N}$ // assemble weight classification // data using selector

return $\mathcal{A}(D^{weighted})$ // train classifier

We will revisit logistic regression!

Supervised adaptation

Goal: learn a classifier f that achieves low expected loss under new distribution $\mathcal{D}^{new}$

Given labeled training data from old distribution

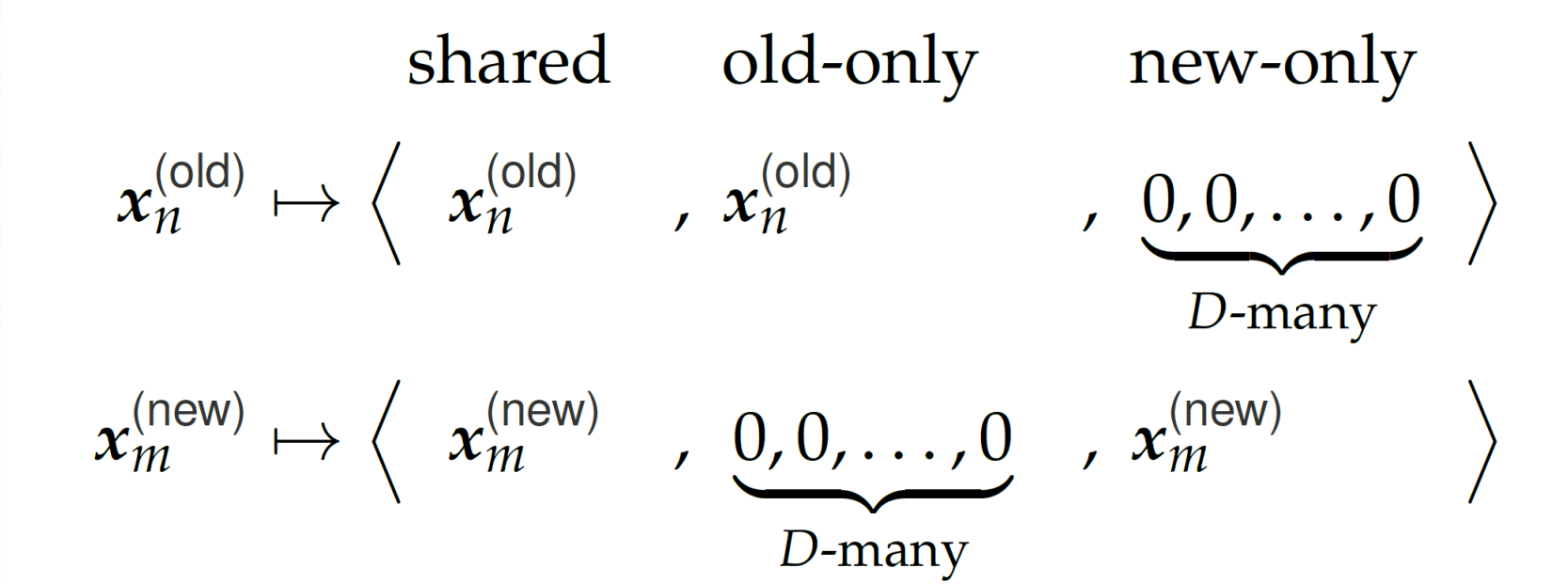
$\mathcal{D}^{\text{old}}$ $\langle \boldsymbol{x}_n^{(\text{old})}, y_n^{(\text{old})} \rangle_{n=1}^{N}$

And labeled examples from new distribution

$\mathcal{D}^{\text{new}}: \langle \boldsymbol{x}_m^{(\text{new})}, y_m^{(\text{new})} \rangle_{m=1}^{M}$

One solution: feature augmentation

Map inputs to a new augmented representation



One solution: feature augmentation

- Transform D_{old} and D_{new} training examples
- Train a classifier on new representations
- Done!

One solution: feature augmentation

- Adding instance weighting might be useful if N >> M

- Most effective when distributions are "not too close but not too far"
  - In practice, always try "old only", "new only", "union of old and new" as well!

Bias and how to deal with it

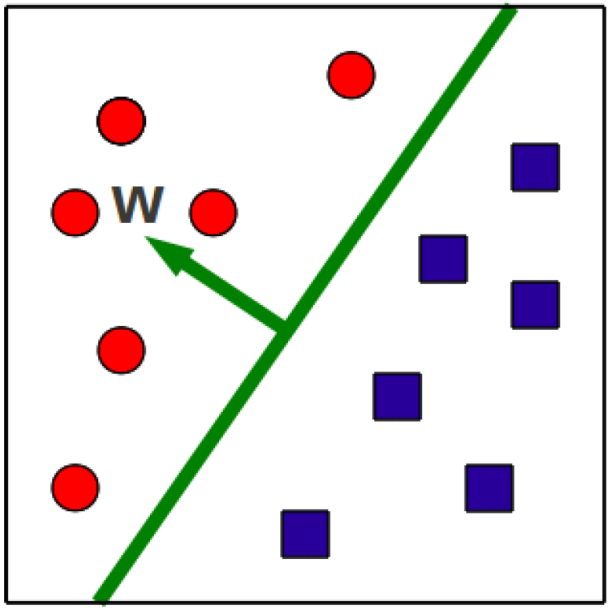
- Train/test mismatch
- Unsupervised adaptation
- Supervised adaptation

Topics

- Linear Models
  - Loss functions
  - Regularization
- Gradient Descent
- Calculus refresher
  - Convexity
  - Gradients

[CIML Chapter 6]

Binary classification via hyperplanes



A classifier is a hyperplane (w,b)

At test time, we check on what side of the hyperplane examples fall

$$\hat{y} = sign(w^T x + b)$$

This is a **linear classifier**

Because the prediction is a linear combination of feature values x

Task: Binary Classification

Given:

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$

Compute: A function f minimizing: $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[f(\boldsymbol{x}) \neq y\right]$

Learning a Linear Classifier as an Optimization Problem

Objective function

Loss function measures how well classifier fits training data

Regularizer prefers solutions that generalize well

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T\mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

$\mathbb{I}(.)$ Indicator function: 1 if (.) is true, 0 otherwise

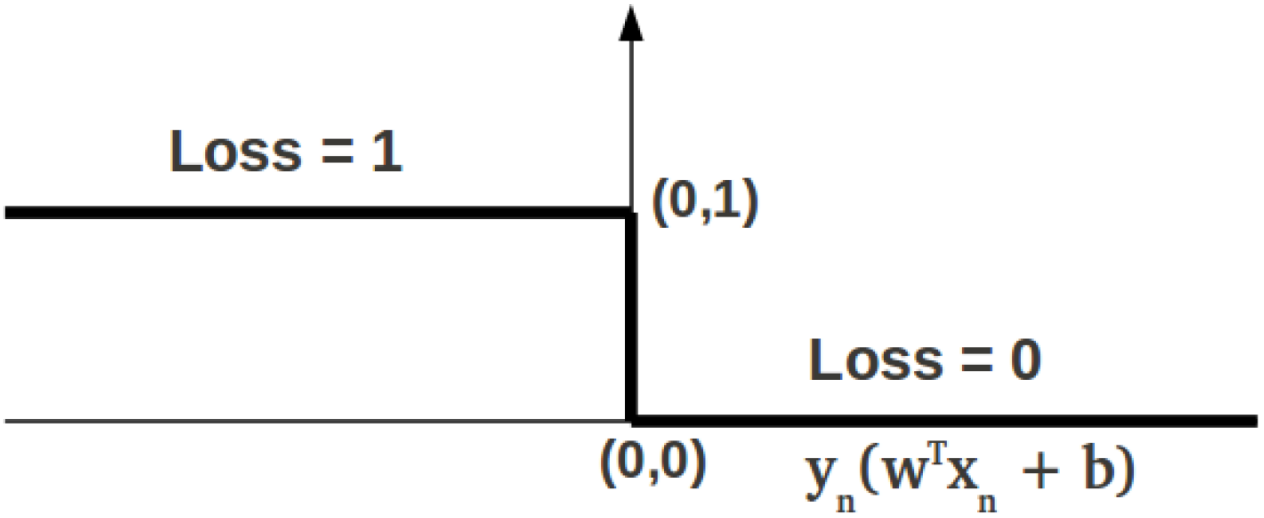
The loss function above is called the 0-1 loss

Learning a Linear Classifier as an Optimization Problem

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T\mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

- **Problem:** The 0-1 loss above is NP-hard to optimize exactly/approximately in general
- **Solution:** Different loss function approximations and regularizers lead to specific algorithms
  (e.g., perceptron, support vector machines, logistic regression, etc.)

The 0-1 Loss



Small changes in w,b can lead to big changes in the loss value

0-1 loss is non-smooth, non-convex

Calculus refresher:

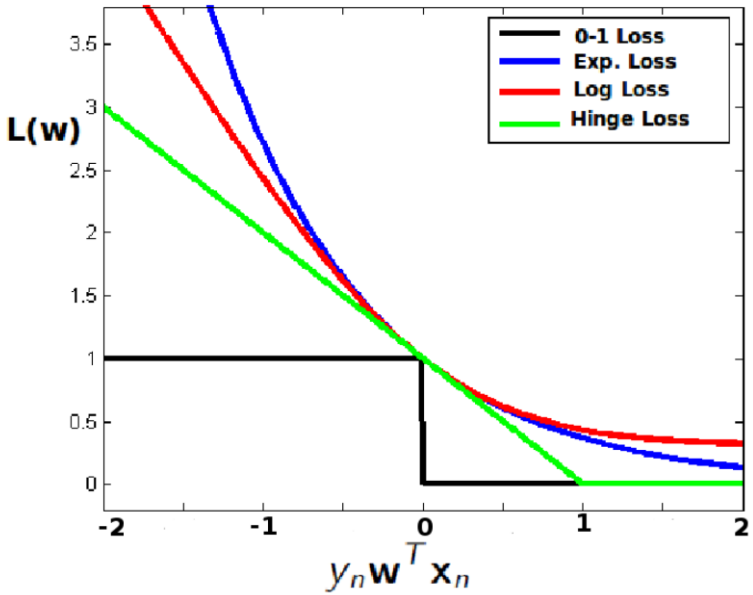
Smooth functions, convex functions

Approximating the 0-1 loss with surrogate loss functions

Examples (with b = 0)

- Hinge loss $[1 - y_n \mathbf{w}^T \mathbf{x}_n]_+ = \max\{0, 1 - y_n \mathbf{w}^T \mathbf{x}_n\}$
- Log loss $\log[1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)]$
- Exponential loss $\exp(-y_n \mathbf{w}^T \mathbf{x}_n)$

All are convex upper-bounds on the 0-1 loss

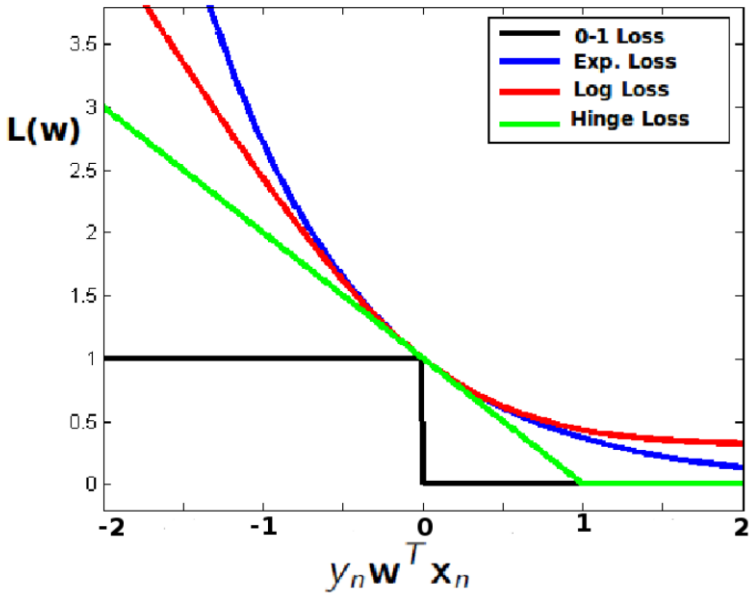


Approximating the 0-1 loss with surrogate loss functions

Examples (with b = 0)

- Hinge loss $[1 - y_n \mathbf{w}^T \mathbf{x}_n]_+ = \max\{0, 1 - y_n \mathbf{w}^T \mathbf{x}_n\}$
- Log loss $\log[1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)]$
- Exponential loss $\exp(-y_n \mathbf{w}^T \mathbf{x}_n)$

Q: Which of these loss functions is not smooth?

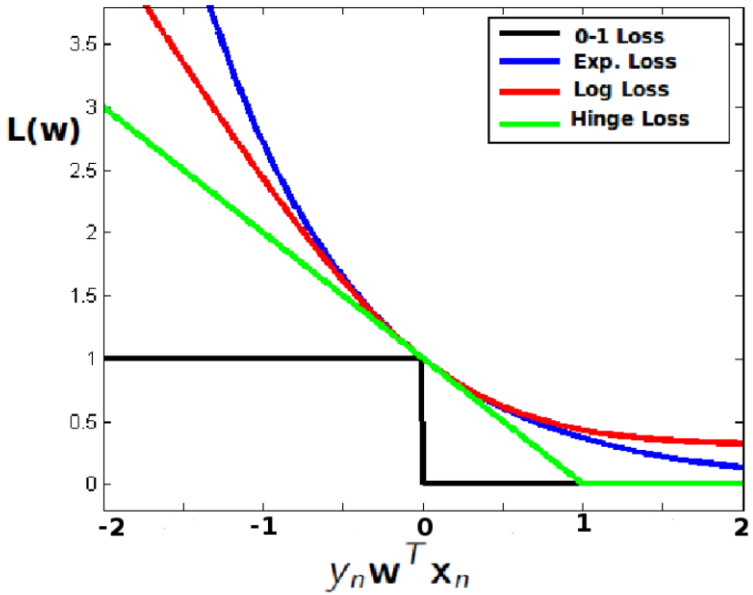


Approximating the 0-1 loss with surrogate loss functions

Examples (with b = 0)

- Hinge loss $[1 - y_n \mathbf{w}^T \mathbf{x}_n]_+ = \max\{0, 1 - y_n \mathbf{w}^T \mathbf{x}_n\}$
- Log loss $\log[1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)]$
- Exponential loss $\exp(-y_n \mathbf{w}^T \mathbf{x}_n)$

Q: Which of these loss functions is most sensitive to outliers?



Casting Linear Classification as an Optimization Problem

Objective function

Loss function measures how well classifier fits training data

Regularizer prefers solutions that generalize well

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T\mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

$\mathbb{I}(.)$ Indicator function: 1 if (.) is true, 0 otherwise

The loss function above is called the 0-1 loss

The regularizer term

Goal: find simple solutions (inductive bias)

Ideally, we want most entries of w to be zero, so prediction depends only on a small number of features.

Formally, we want to minimize:

$$R^{cnt}(\mathbf{w}, b) = \sum_{d=1}^{D} \mathbb{I}(w_d \neq 0)$$

That's NP-hard, so we use approximations instead.

E.g., we encourage w_d's to be small

Norm-based Regularizers

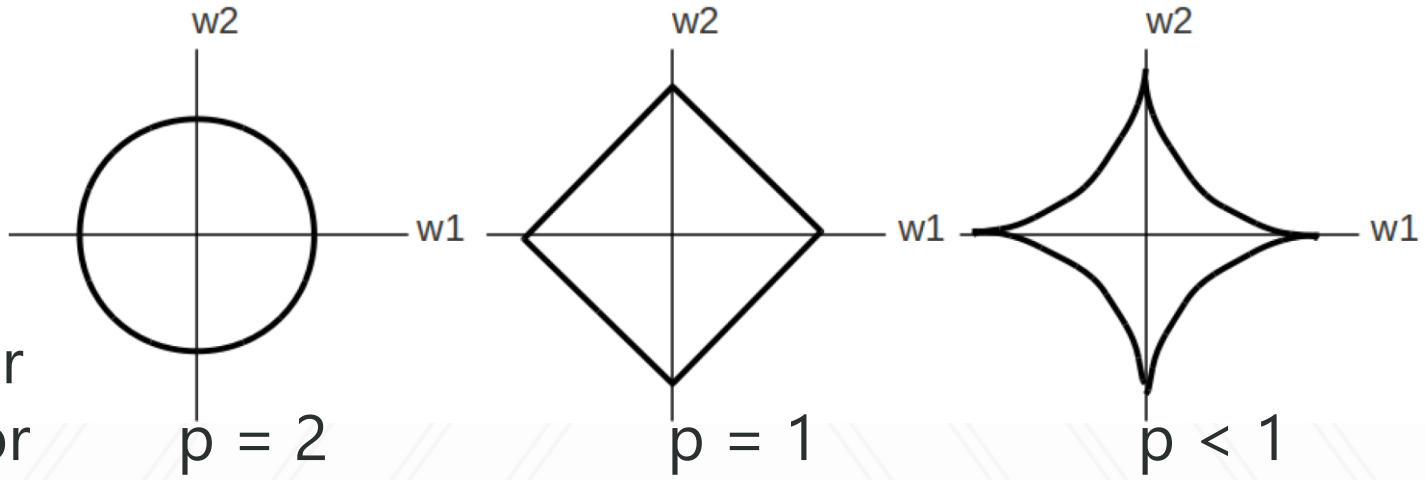
l_p norms can be used as regularizers

$$||\mathbf{w}||_2^2 = \sum_{d=1}^{D} w_d^2$$

$$||\mathbf{w}||_1 = \sum_{d=1}^{D} |w_d|$$

$$||\mathbf{w}||_p = \left(\sum_{d=1}^{D} w_d^p\right)^{1/p}$$

Contour plots for



Norm-based Regularizers

l_p norms can be used as regularizers

Smaller p favors sparse vectors w

i.e. most entries of w are close or equal to 0

l_2 norm: convex, smooth, easy to optimize

l_1 norm: encourages sparse w, convex, but not smooth at axis points

$p < 1$: norm becomes non convex and hard to optimize

Casting Linear Classification as an Optimization Problem

Objective function

Loss function measures how well classifier fits training data

Regularizer prefers solutions that generalize well

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T\mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

$\mathbb{I}(.)$ Indicator function: 1 if (.) is true, 0 otherwise

The loss function above is called the 0-1 loss

What is the perceptron optimizing?

Algorithm 5 PERCEPTRONTRAIN(**D**, *MaxIter*)

```
w_d ← 0, for all d = 1 ... D                          // initialize weights
b ← 0                                                  // initialize bias
for iter = 1 ... MaxIter do
    for all (x,y) ∈ D do
        a ← Σ_{d=1}^{D} w_d x_d + b                    // compute activation for this example
        if ya ≤ 0 then
            w_d ← w_d + y x_d, for all d = 1 ... D     // update weights
            b ← b + y                                  // update bias
        end if
    end for
end for
return w_0, w_1, ..., w_D, b
```

Loss function is a variant of the hinge loss

$$\max\{0, -y_n(\mathbf{w}^T\mathbf{x}_n + b)\}$$

Recap: Linear Models

General framework for binary classification

Cast learning as optimization problem

Optimization objective combines 2 terms

- loss function: measures how well classifier fits training data
- Regularizer: measures how simple classifier is

- Does not assume data is linearly separable

Lets us separate model definition from training algorithm

Calculus refresher:
Gradients

Gradient descent

A general solution for our optimization problem

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T\mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

Idea: take iterative steps to update parameters in the direction of the gradient

Gradient descent algorithm

Algorithm 22 GRADIENTDESCENT($\mathcal{F}$, K, η_1, ...)

1: $z^{(0)} \leftarrow \langle 0, 0, \ldots, 0 \rangle$ // initialize variable we are optimizing
2: **for** $k = 1 \ldots K$ **do**
3: $g^{(k)} \leftarrow \nabla_z \mathcal{F}|_{z^{(k-1)}}$ // compute gradient at current location
4: $z^{(k)} \leftarrow z^{(k-1)} - \eta^{(k)} g^{(k)}$ // take a step down the gradient
5: **end for**
6: **return** $z^{(K)}$

Recap: Linear Models

General framework for binary classification

Cast learning as optimization problem

Optimization objective combines 2 terms

- loss function: measures how well classifier fits training data
- Regularizer: measures how simple classifier is

- Does not assume data is linearly separable

Lets us separate model definition from training algorithm **(Gradient Descent)**

UNIVERSITY OF
MARYLAND

Furong Huang
4124 IRB, College Park, MD 20740
301.405.8010 / furongh@umd.edu