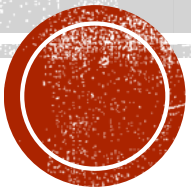
LINEAR MODELS



CS 412 Introduction to Machine Learning

Prof. Zheleva

February 15, 2017

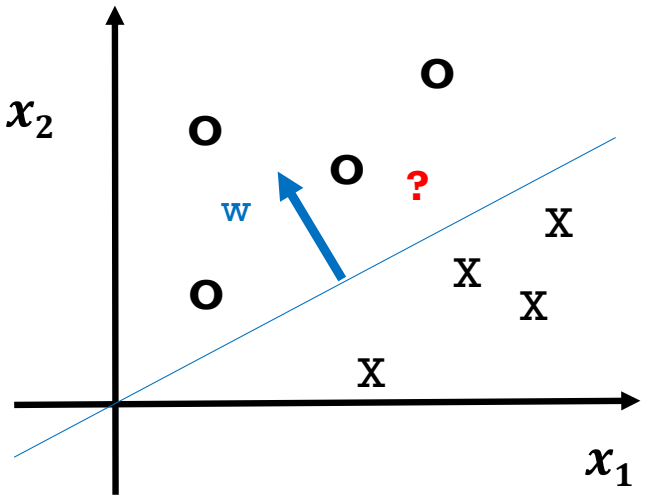
Reading Assignment: CIML: 7

BINARY CLASSIFICATION VIA HYPERPLANES

- A classifier is a hyperplane that separates positive from negative examples (**w**,b)
- At test time, we check on what side of the hyperplane examples fall

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

- This is a linear classifier model
  - Because the prediction is a linear combination of feature values x
  - Perceptron is one example



TASK: BINARY CLASSIFICATION

Given:

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$
3. A training set D sampled from $\mathcal{D}$

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

EXAMPLE ALGORITHM FOR LINEAR CLASSIFICATION: PERCEPTRON

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

1: $w_d \leftarrow 0$, for all $d = 1 \ldots D$ // initialize weights
2: $b \leftarrow 0$ // initialize bias
3: **for** $iter = 1 \ldots MaxIter$ **do**
4: **for all** $(x,y) \in \mathbf{D}$ **do**
5: $a \leftarrow \sum_{d=1}^{D} w_d \, x_d + b$ // compute activation for this example
6: **if** $ya \leq 0$ **then**
7: $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ // update weights
8: $b \leftarrow b + y$ // update bias
9: **end if**
10: **end for**
11: **end for**
12: **return** $w_0, w_1, \ldots, w_D, b$

LEARNING A LINEAR CLASSIFIER AS AN OPTIMIZATION PROBLEM

- Not all data is linearly separable
- Goal: minimize the number of misclassified examples

Indicator function: 1 if (.) is true, 0 otherwise

$$\min_{w,b} \sum_{n} \mathbf{1}[y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 0]$$

Objective function
In this case: simple 0/1 loss

Variables over which we are optimizing the objective function

LEARNING A LINEAR CLASSIFIER AS AN OPTIMIZATION PROBLEM

- If linearly separable, efficient algorithms exist
- If not linearly separable, problem is NP-hard
- What about approximate optimization (small constant worse than optimal)?
  - Problem is still NP-hard
- This is just optimizing training error!
  - We care about generalization

LEARNING A LINEAR CLASSIFIER AS AN OPTIMIZATION PROBLEM

- Structural risk minimization framework
  - Tradeoff between low training error and a solution that is simple

$$\min_{w,b} \sum_n \mathbf{1}[y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 0] + \lambda R(\boldsymbol{w}, b)$$

Variables over which we are optimizing the objective function

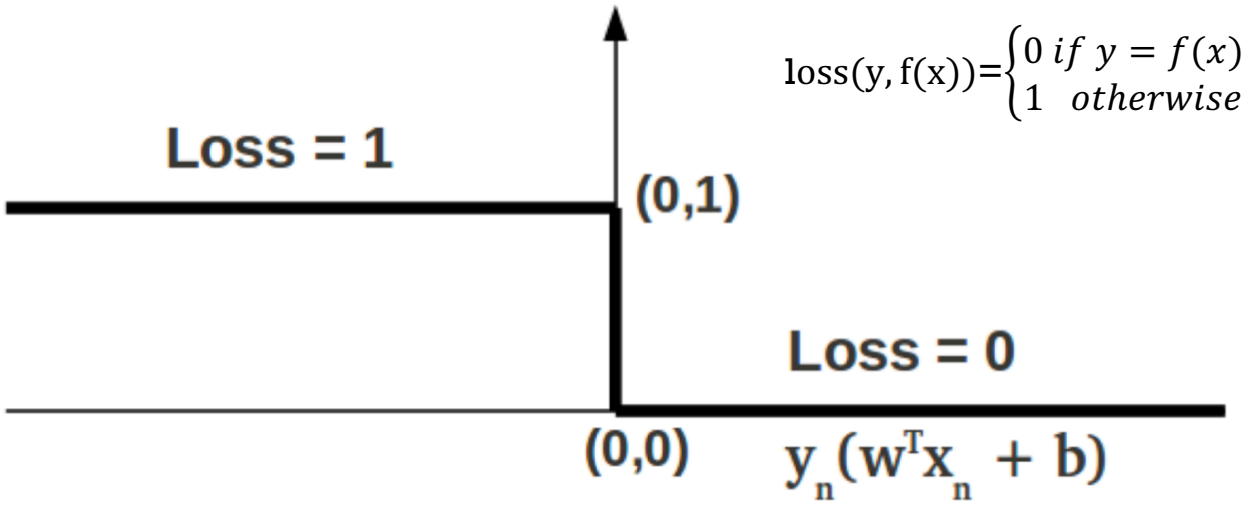
Objective function
In this case: simple 0/1 loss

Regularizing function
Helps with avoiding overfitting
Penalizes complex functions
λ is a hyperparameter

KEY QUESTIONS

- How can we adjust the optimization problem so that there *are* efficient algorithms for solving it?
- What are good regularizers $R(\boldsymbol{w}, b)$ for hyperplanes?
- Assuming we can adjust the optimization problem appropriately, what algorithms exist for efficiently solving this regularized optimization problem?

THE 0/1 LOSS FUNCTION



$$\text{loss}(y, f(x)) = \begin{cases} 0 \; if \; y = f(x) \\ 1 \quad otherwise \end{cases}$$

- Small changes in w,b can lead to big changes in the loss
- 0/1 loss is non-smooth, non-convex

APPROXIMATING THE 0/1 LOSS WITH CONVEX SURROGATE LOSS FUNCTIONS

- Examples (with b=0)

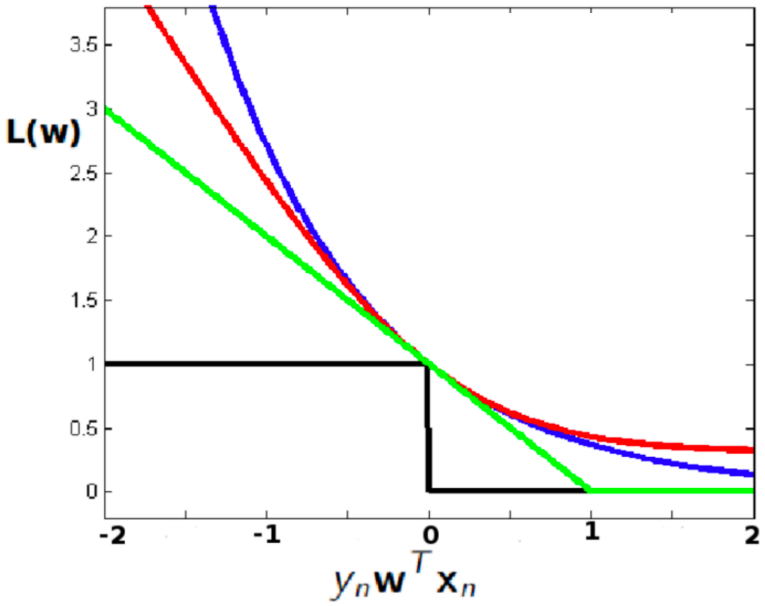
$$\text{Zero/one:} \quad \ell^{(0/1)}(y,\hat{y}) = \mathbf{1}[y\hat{y} \leq 0]$$

$$\text{Hinge:} \quad \ell^{(\text{hin})}(y,\hat{y}) = \max\{0, 1 - y\hat{y}\}$$

$$\text{Logistic:} \quad \ell^{(\log)}(y,\hat{y}) = \frac{1}{\log 2} \log\left(1 + \exp[-y\hat{y}]\right)$$

$$\text{Exponential:} \quad \ell^{(\exp)}(y,\hat{y}) = \exp[-y\hat{y}]$$

$$\text{Squared:} \quad \ell^{(\text{sqr})}(y,\hat{y}) = (y - \hat{y})^2$$



- All are convex upper bounds on 0/1 loss
- Q: Which curve is which function?

APPROXIMATING THE 0/1 LOSS WITH CONVEX SURROGATE LOSS FUNCTIONS

- Examples (with b=0)

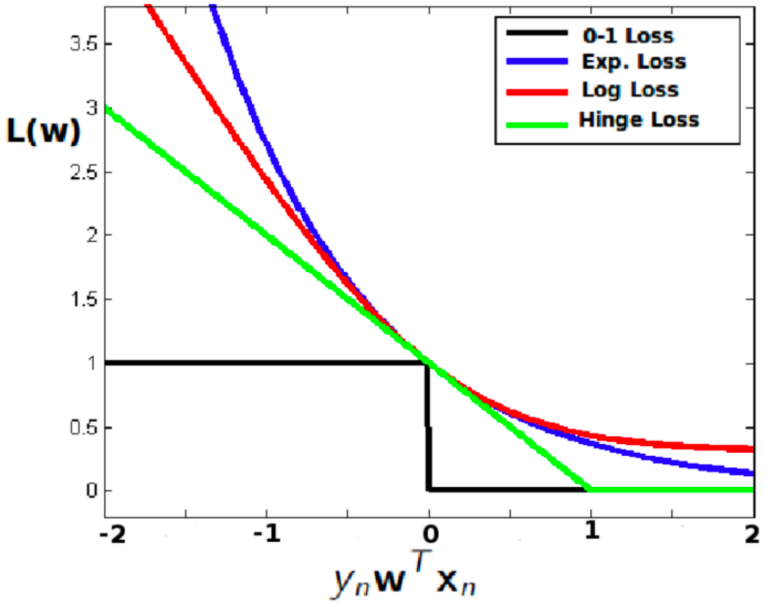
$$\text{Zero/one:} \quad \ell^{(0/1)}(y,\hat{y}) = \mathbf{1}[y\hat{y} \leq 0]$$

$$\text{Hinge:} \quad \ell^{(\text{hin})}(y,\hat{y}) = \max\{0, 1 - y\hat{y}\}$$

$$\text{Logistic:} \quad \ell^{(\log)}(y,\hat{y}) = \frac{1}{\log 2} \log\left(1 + \exp[-y\hat{y}]\right)$$

$$\text{Exponential:} \quad \ell^{(\exp)}(y,\hat{y}) = \exp[-y\hat{y}]$$

$$\text{Squared:} \quad \ell^{(\text{sqr})}(y,\hat{y}) = (y - \hat{y})^2$$



- All are convex upper bounds on 0/1 loss
- Q: Which of these functions is not smooth?

APPROXIMATING THE 0/1 LOSS WITH CONVEX SURROGATE LOSS FUNCTIONS

- Examples (with b=0)

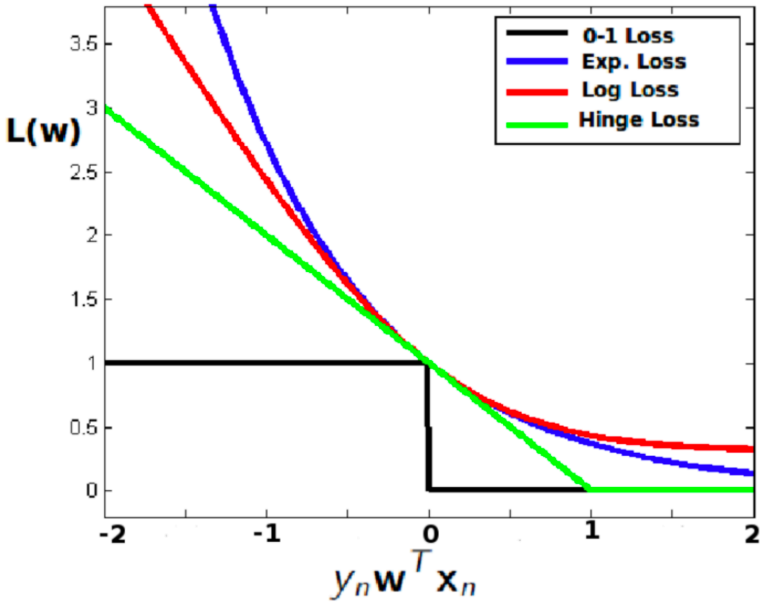
$$\text{Zero/one:} \quad \ell^{(0/1)}(y,\hat{y}) = \mathbf{1}[y\hat{y} \leq 0]$$

$$\text{Hinge:} \quad \ell^{(\text{hin})}(y,\hat{y}) = \max\{0, 1 - y\hat{y}\}$$

$$\text{Logistic:} \quad \ell^{(\log)}(y,\hat{y}) = \frac{1}{\log 2} \log\left(1 + \exp[-y\hat{y}]\right)$$

$$\text{Exponential:} \quad \ell^{(\exp)}(y,\hat{y}) = \exp[-y\hat{y}]$$

$$\text{Squared:} \quad \ell^{(\text{sqr})}(y,\hat{y}) = (y - \hat{y})^2$$



- All are convex upper bounds on 0/1 loss
- Q: Which function is most sensitive to outliers?

LEARNING A LINEAR CLASSIFIER AS AN OPTIMIZATION PROBLEM

- Structural risk minimization framework
  - Tradeoff between low training error and a solution that is simple

$$\min_{\boldsymbol{w},b} L(\boldsymbol{w},b) = \min_{\boldsymbol{w},b}\left(\sum_{n=1}^{N} l(y_n, \boldsymbol{w}\cdot x_n + b) + \lambda R(\boldsymbol{w},b)\right)$$

Objective function

Loss function
Measures how well classifier fits training data

Regularizing function
Helps with avoiding overfitting
Penalizes complex functions
λ is a hyperparameter

Variables over which we are optimizing the objective function

THE REGULARIZER TERM

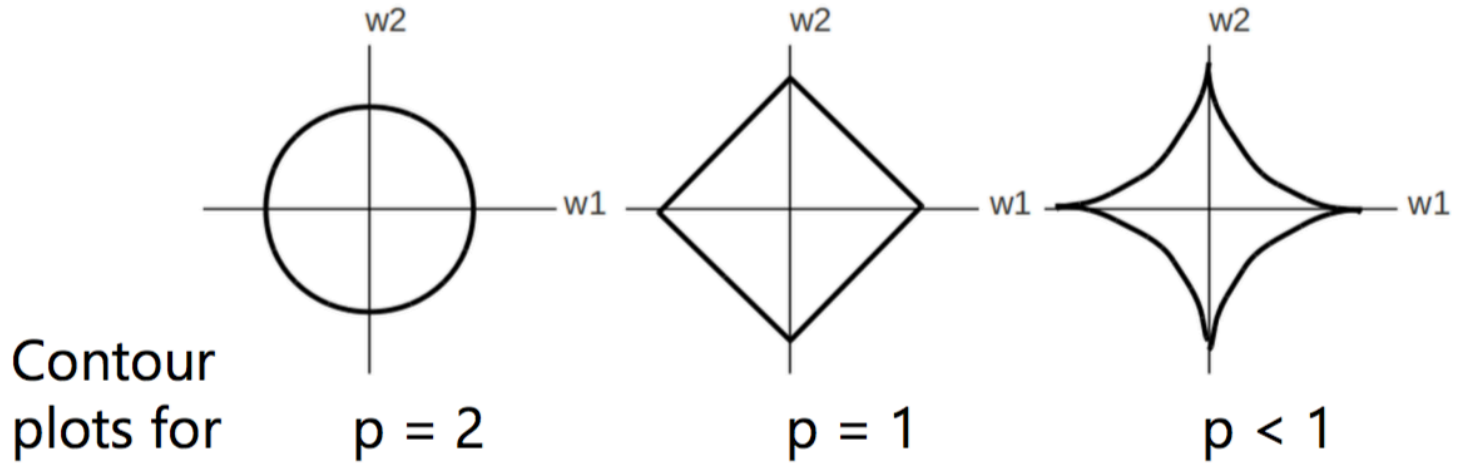
- Goal: find simple solutions (low w_d's, type of inductive bias)
- Example of simple solution
  - If most of w elements are zero, prediction depends only on a small number of features.
  - Formally, we want to minimize:

$$R^{(\text{cnt})}(\boldsymbol{w}, b) = \sum_d \mathbf{1}[x_d \neq 0]$$

  - 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
  - l_2 norm: $||\boldsymbol{w}||_2 = \sqrt{\sum_i w_i^2}$ where i is the dimension
  - l_1 norm: $||\boldsymbol{w}||_1 = \sum_i |w_i|$
  - l_p norm: $||\boldsymbol{w}||_p = (\sum_i w_i^p)^{\frac{1}{p}}$



NORM-BASED REGULARIZERS

- l_p norms can be used as regularizers
- Smaller p favors sparser 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

LEARNING A LINEAR CLASSIFIER AS AN OPTIMIZATION PROBLEM

- Structural risk minimization framework
  - Tradeoff between low training error and a solution that is simple

$$\min_{\boldsymbol{w},b} L(\boldsymbol{w},b) = \min_{\boldsymbol{w},b}\left(\sum_{n=1}^{N} l(y_n, \boldsymbol{w}\cdot x_n + b) + \lambda R(\boldsymbol{w},b)\right)$$

Objective function

Loss function
Measures how well classifier fits training data

Regularizing function
Helps with avoiding overfitting
Penalizes complex functions
λ is a hyperparameter

Variables over which we are optimizing the objective function

RECAP: LINEAR MODELS

- General framework for binary classification
- Cast learning as optimization problem
- Optimization objective combines two 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)**

GRADIENT DESCENT

- A general solution for our optimization problem

$$\min_{\boldsymbol{w},b} L(\boldsymbol{w},b) = \min_{\boldsymbol{w},b}\left(\sum_{n=1}^{N} l(y_n, \boldsymbol{w}\cdot x_n + b) + \lambda R(\boldsymbol{w},b)\right)$$

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

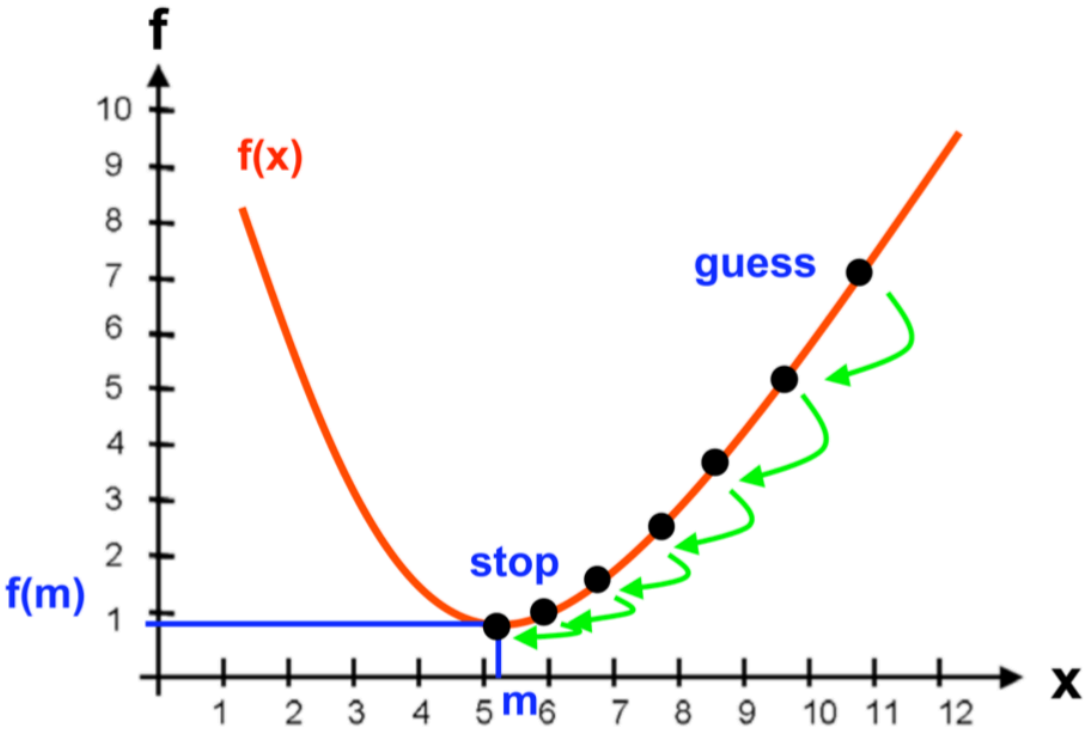
GRADIENT DESCENT ALGORITHM

$\mathcal{F}$: Objective function to minimize
K: Number of steps
η_1: first step size

Algorithm 21 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: $g^{(k)} \leftarrow \nabla_z \mathcal{F}\big|_{z^{(k-1)}}$ // 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** $z^{(K)}$

ILLUSTRATING GRADIENT DESCENT IN 1-DIMENSIONAL CASE



GRADIENT DESCENT

- Two questions:
  - When to stop?
    - When the gradient gets close to zero
    - When the objective stops changing much
    - When the parameters stop changing much early
    - When performance on held-out development set plateaus
  - How to choose the step size?
    - Start with large steps, then take smaller steps

NOW LET'S CALCULATE GRADIENTS FOR MULTIVARIATE OBJECTIVES

- Consider the following learning objective
  - Loss function: Logistic loss
  - Regularizer: Squared l_2 norm

$$\min_{\boldsymbol{w},b} L(\boldsymbol{w}, b) = \min_{\boldsymbol{w},b} \left(\sum_{n=1}^{N} exp(-y_n(\boldsymbol{w} \cdot x_n + b)) + \frac{\lambda}{2} \|w\|_2^2 \right)$$

- What do we need to do to run gradient descent?

1) DERIVATIVE WITH RESPECT TO B

$$\min_{\boldsymbol{w},b} L(\boldsymbol{w},b) = \min_{\boldsymbol{w},b}\left(\sum_{n=1}^{N} exp(-y_n(\boldsymbol{w}\cdot x_n + b)) + \frac{\lambda}{2}\|w\|_2^2\right)$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial}{\partial b}\sum_n \exp\left[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\right] + \frac{\partial}{\partial b}\frac{\lambda}{2}||\boldsymbol{w}||^2 \tag{7.14}$$

$$= \sum_n \frac{\partial}{\partial b}\exp\left[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\right] + 0 \tag{7.15}$$

$$= \sum_n \left(\frac{\partial}{\partial b} - y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\right)\exp\left[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\right] \tag{7.16}$$

$$= -\sum_n y_n \exp\left[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\right] \tag{7.17}$$

2) DERIVATIVE WITH RESPECT TO W

$$\min_{\boldsymbol{w},b} L(\boldsymbol{w},b) = \min_{\boldsymbol{w},b}\Big(\sum_{n=1}^{N} exp(-y_n(\boldsymbol{w}\cdot x_n + b)) + \frac{\lambda}{2}\|w\|_2^2\Big)$$

$$\nabla_w \mathcal{L} = \nabla_w \sum_n \exp\big[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\big] + \nabla_w \frac{\lambda}{2}||\boldsymbol{w}||^2 \tag{7.18}$$

$$= \sum_n (\nabla_w - y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)) \exp\big[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\big] + \lambda\boldsymbol{w}$$

(7.19)

$$= -\sum_n y_n\boldsymbol{x}_n \exp\big[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\big] + \lambda\boldsymbol{w} \tag{7.20}$$

SUBGRADIENTS

- Problem: some objective functions are not differentiable everywhere
  - Hinge loss (around 1)
  - l_1 norm (when wd=0)
- Solution: subgradient optimization
  - Let's ignore the problem, and just try to apply gradient descent anyway!!
  - We will just differentiate by parts…

SUMMARY

- Gradient descent
  - A generic algorithm to minimize objective functions
  - Works well as long as functions are well behaved (i.e. convex)
  - Subgradient descent can be used at points where derivative is not defined
  - Choice of step size is important
- Optional: can we do better?
  - For some objectives, we can find closed form solutions
    - see CIML 7.6

ANNOUNCEMENTS: HW2

"**What to Submit:** You will hand in all of the python files listed above together with your notebook **hw2.ipynb** as a single zip file **h2.zip** on Gradescope under *Homework 2*. The programming part constitutes 60% of the grade for this homework. You also need to answer the questions denoted by **WU#** (and a kitten) in this notebook which are the other 40% of your homework grade. When you are done, you should export **hw2.ipynb** with your answers as a PDF file **hw2WrittenPart.pdf** and upload the PDF file to Gradescope under *Homework 2 - Written Part*. Your entire homework will be considered late if any of these parts are submitted late."

If you don't see this kitten in your notebook, you are working on the wrong notebook version and need to download HW2.zip again.
If you don't submit the written part in the required format, you are risking losing 40% of the HW grade.

ACKNOWLEDGEMENTS

- These slides use materials by Marine Carpuat