Machine Learning Basics Pt 2

Can Serif Mekik
PhD Candidate
Rensselaer Polytechnic Institute

April 28, 2022

CDSI
McGill Faculty of Science

McGill University • Co-Curricular Record
APPROVED

Are you interested in machine learning? If you are looking for a place to get started, then please join us for this workshop where we will cover basic concepts in machine learning theory and briefly survey some popular machine learning models.

At the end of this workshop (parts 1+2), you will be able to:
> Identify common machine learning problems and models.
> Understand basic concepts in machine learning theory and practice
> Create and run simple machine learning models using the scikit-learn Python library

Pre-requisites? Working knowledge of Python, linear algebra, calculus, and probability theory.

Today

1. ~~Welcome~~
2. Review
3. Perceptrons, Revisited
4. Stochastic Gradient Descent
5. A Brief Look at Learning Theory
6. Neural Networks
7. Support Vector Machines

Software & Materials

scikit-learn – Python ML library

- Relatively Mature
- Versatile
- Available as part of the Anaconda stack

Anaconda: https://www.anaconda.com/

Materials: https://github.com/cmekik/CDSI-MLB

Review: Dataset

Handwritten digit classification

- Classic task for computer vision & neural networks
- ‘digits’ dataset included in scikit-learn

Review: Topics

- Motivation for ML
- ML Cycle
- Perceptron
- Underfitting, Overfitting, & Regularization
- Features & Feature Transformations

Correction: The Perceptron Model

input

$$\cancel{h(\mathbf{x_i}|\mathbf{w}) = sign\left(\sum_{j=1}^{N} w_{ji} x_i + b\right)}$$

parameter intercept

~~Accuracy~~
Error Rate

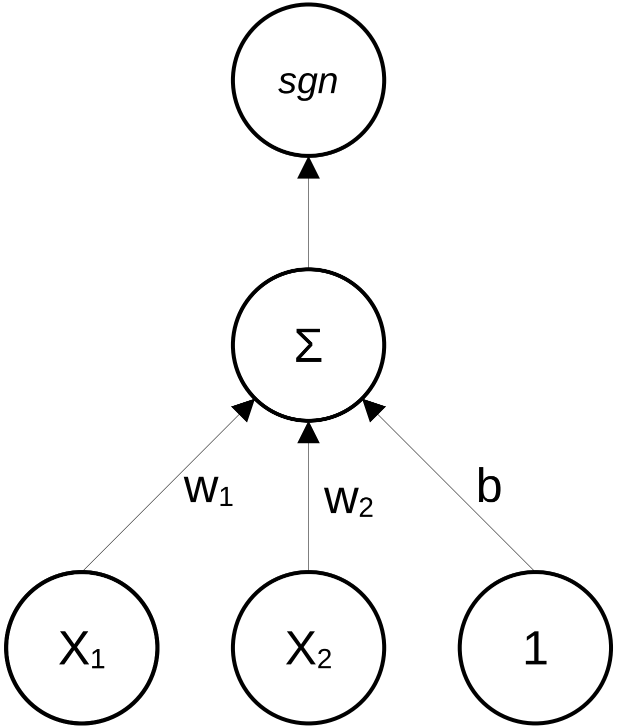
$$\frac{1}{N}\sum_{i=1}^{N} [\![h(\mathbf{x_i}|\mathbf{w}) \neq y_i]\!]$$

input

$$\boxed{h(\mathbf{x_i}|\mathbf{w}) = sign\left(\sum_{j=1}^{N} w_j x_{ij} + b\right)}$$

parameter intercept

Training Perceptrons



<u>Algorithm Outline</u>

- Initialize randomly
- Unless satisfied:
  - Compute error on sample
  - Adjust params to minimize error
  - Repeat

Your stopping criterion is essential!

(Stochastic) Gradient Descent

Data

Cost function

iteration

$$w_i(t+1) = w_i(t) - \eta \frac{\partial C(\mathbf{x}(t), \mathbf{y}(t) \mid \mathbf{w}(t))}{\partial w_i(t)}$$

parameter

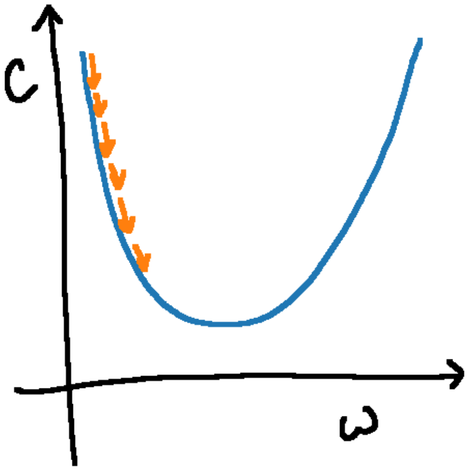
Learning rate

Minimize cost function by incrementally adjusting each parameter in the direction opposite the gradient

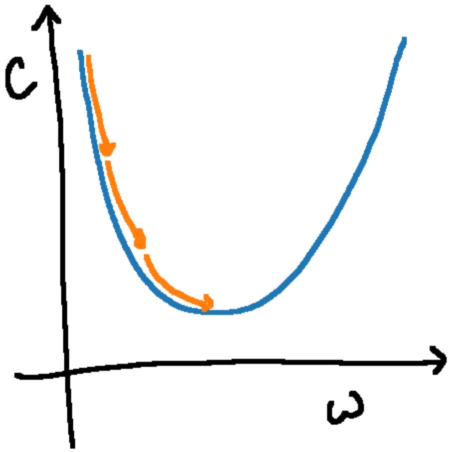
Cost function may include regularization term, if used

Setting the Learning Rate

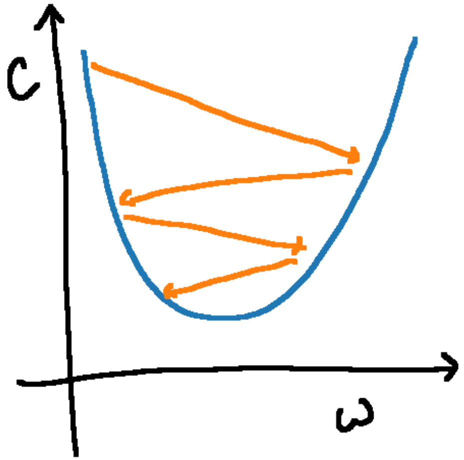
Too small



Okay



Too big



SGD for Perceptrons

$$f(\boldsymbol{x_i}) := b + \sum_{j=1}^{D} w_j x_{ij}$$
$$C(\boldsymbol{x_i}, y_i | \boldsymbol{w}) = max(0, 1 - y_i f(x_i))$$
$$0 \leqslant [\![h(\boldsymbol{x_i}|\boldsymbol{w}) \neq y_i]\!] \leqslant C(\boldsymbol{x_i}, y_i | \boldsymbol{w})$$

Sign function not differentiable!

So, use a trick (the hinge loss function)

SGD for Perceptrons (cont.)

$$\frac{\partial C(\boldsymbol{x_i}, y_i | \boldsymbol{w})}{\partial w_j} = \frac{\partial C(\boldsymbol{x_i}, y_i | \boldsymbol{w})}{\partial f(\boldsymbol{x_i} | \boldsymbol{w})} \frac{\partial f(\boldsymbol{x_i} | \boldsymbol{w})}{\partial w_j}$$

$$\frac{\partial C(\boldsymbol{x_i}, y_i | \boldsymbol{w})}{\partial w_j} = \begin{cases} 0 & \text{if } 1 < y_i f(\boldsymbol{x_i}) \\ -y_i x_{ij} & \text{otherwise} \end{cases}$$

Learning rule for each param can be derived using the chain rule

This is the basic idea behind backpropagation

Theory: Feasibility of Learning

- Can we make out-of-sample error close enough to in-sample error?
- Can we make in-sample error small enough?

Theory: Generalization Error

In sample error

Out of sample error

Size of hypothesis set

$$\Pr\left[\left|E_{\text{in}}(h)-E_{\text{out}}(h)\right|>\epsilon\right]\leqslant 2M\exp\left(-2\epsilon^2 N\right)$$

Hoeffding Inequality (for binary classification)

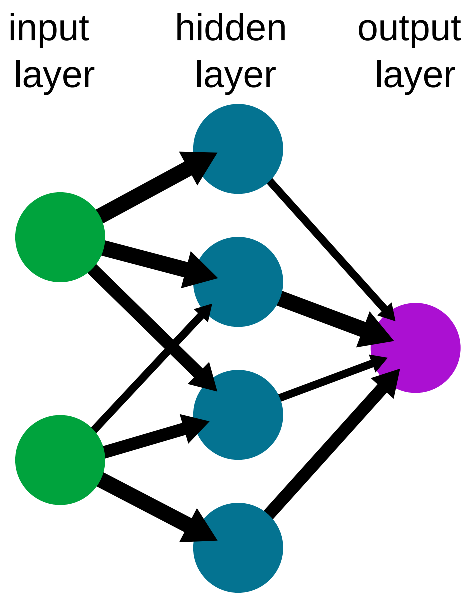
tolerance

Sample size

Cannot guarantee generalization, but can give probabilistic bound
Bound gets looser with smaller tolerance, larger hypothesis set
Bound gets tighter with more N
This bound is loose, can obtain tighter bounds with more sophisticated methods (e.g. using VC dimensions)

From Perceptrons to Deep Learning

A simple neural network



User:Wiso, Public domain, via Wikimedia Commons

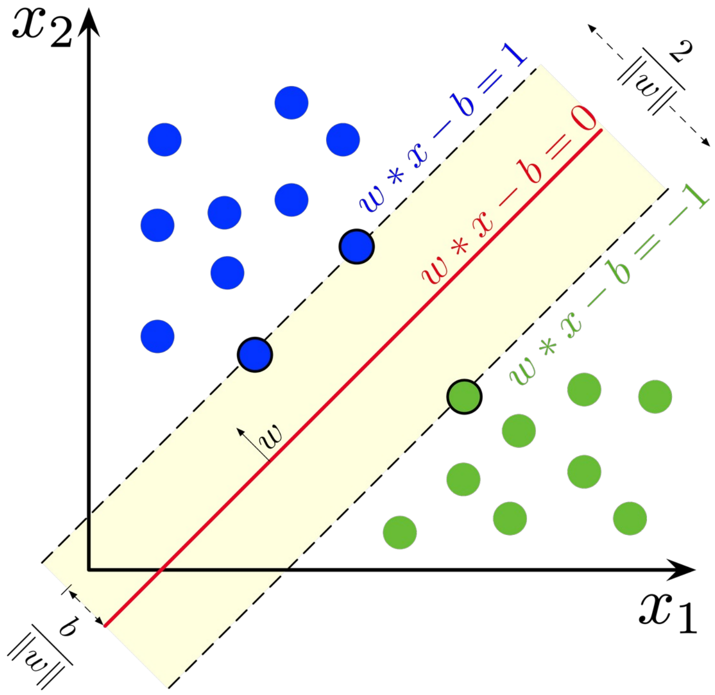
It’s pretty easy to create multi-output perceptrons

Using a differentiable non-linearity, we can propagate errors through perceptron outputs nodes

Propagation works just the same if we ‘stack’ perceptrons on top of each other

This is the basic idea behind a ‘Multilayer Perceptron’, the simplest form of deep neural networks

From Perceptrons to SVMs



Larhmam, CC BY-SA 4.0, via Wikimedia Commons

Basic concept is to obtain the widest separating margin

This optimizes generalization performance

Development more driven by ML theory

Can use non-linear transforms for sophisticated boundaries

Can use soft-margin methods for non-separable data

Further Reading

Learning from data: A short course

http://amlbook.com/index.html