Lecture Notes for

Machine Learning in Python

⟦👨‍🏫, 👨‍💻, 🐍, 👨‍🔬⟧

Professor Eric Larson

Optimization Techniques for Logistic Regression

Class Logistics and Agenda

- Logistics: grading, guest lecture next week
- Agenda
  - Finish Logistic Regression
  - Numerical Optimization Techniques
    - Types of Optimization
    - Programming the Optimization
- **Whirlwind Lecture Alert**
  - Get an intuition, program it, maybe you don't follow every mathematical concept in lecture
  - But you know how to approach it outside lecture

Class Overview, by topic

Table Data Visualization
Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing
Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression
Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.
Numpy
Detailed mathematics for NN operations

Wide and Deep Networks | **Convolutional Networks** | **Recurrent Networks**
Keras, Tensorflow
Intuition, Detailed implement.

Ethics in Language Models

ConceptNet
Case studies

Objective Function: $l(\mathbf{w}) = \sum_i y^{(i)} \ln\left(g(\mathbf{x}^{(i)} \cdot \mathbf{w})\right) + (1 - y^{(i)}) \ln\left(1 - g(\mathbf{x}^{(i)} \cdot \mathbf{w})\right)$

$$\underbrace{w_j}_{\text{new value}} \leftarrow \underbrace{w_j}_{\text{old value}} + \underbrace{\frac{\eta}{M}}_{\text{step}} \underbrace{\sum_{i=1}^{M} (y^{(i)} - g(\mathbf{x}^{(i)} \cdot \mathbf{w}))x_j^{(i)}}_{\text{gradient}}$$

now $\mathbf{x}$, $\mathbf{w}$ are vectors

$$\underbrace{\mathbf{w}}_{\text{new vect}} \leftarrow \underbrace{\mathbf{w}}_{\text{old vect}} + \frac{\eta}{M} \sum_{i=1}^{M} (y^{(i)} - g(\mathbf{x}^{(i)} \cdot \mathbf{w})) \cdot \mathbf{x}^{(i)}$$

weighted sum of $\mathbf{x}$

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot \text{mean}(\mathbf{y}_{diff} \odot \mathbf{X})_{columns}$$

$$\mathbf{y}_{diff} = \begin{bmatrix} y^{(1)} - g(\mathbf{x}^{(1)} \cdot \mathbf{w}) \\ y^{(2)} - g(\mathbf{x}^{(2)} \cdot \mathbf{w}) \\ \vdots \\ y^{(M)} - g(\mathbf{x}^{(M)} \cdot \mathbf{w}) \end{bmatrix} = \begin{bmatrix} y_{diff}^{(1)} \\ y_{diff}^{(2)} \\ \vdots \\ y_{diff}^{(M)} \end{bmatrix}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot \text{mean}\left(\begin{bmatrix} y_{diff}^{(1)} \\ y_{diff}^{(2)} \\ \vdots \\ y_{diff}^{(M)} \end{bmatrix} \odot \begin{bmatrix} \leftarrow & \mathbf{x}^{(1)} & \rightarrow \\ \leftarrow & \mathbf{x}^{(2)} & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{x}^{(M)} & \rightarrow \end{bmatrix} \right)$$

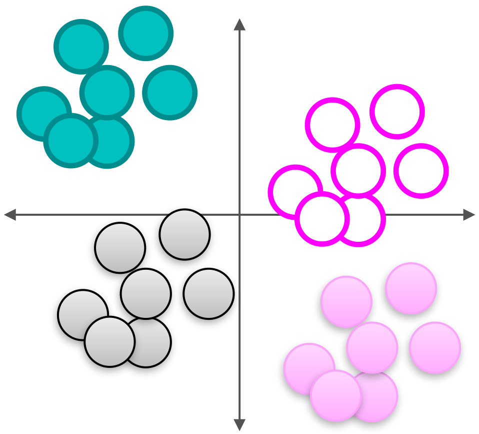
mean along each column

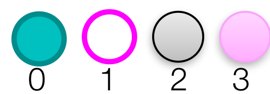
$$\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot \text{mean}\left(\begin{bmatrix} (\leftarrow & \mathbf{x}^{(1)} & \rightarrow) \cdot y_{diff}^{(1)} \\ (\leftarrow & \mathbf{x}^{(2)} & \rightarrow) \cdot y_{diff}^{(2)} \\ & \vdots & \\ (\leftarrow & \mathbf{x}^{(M)} & \rightarrow) \cdot y_{diff}^{(M)} \end{bmatrix} \right)$$

mean along each column

Review: One Versus Rest

Multi-class formulation

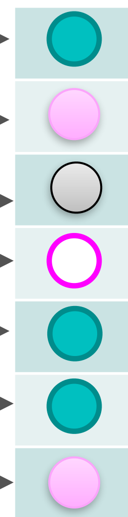


classes[]

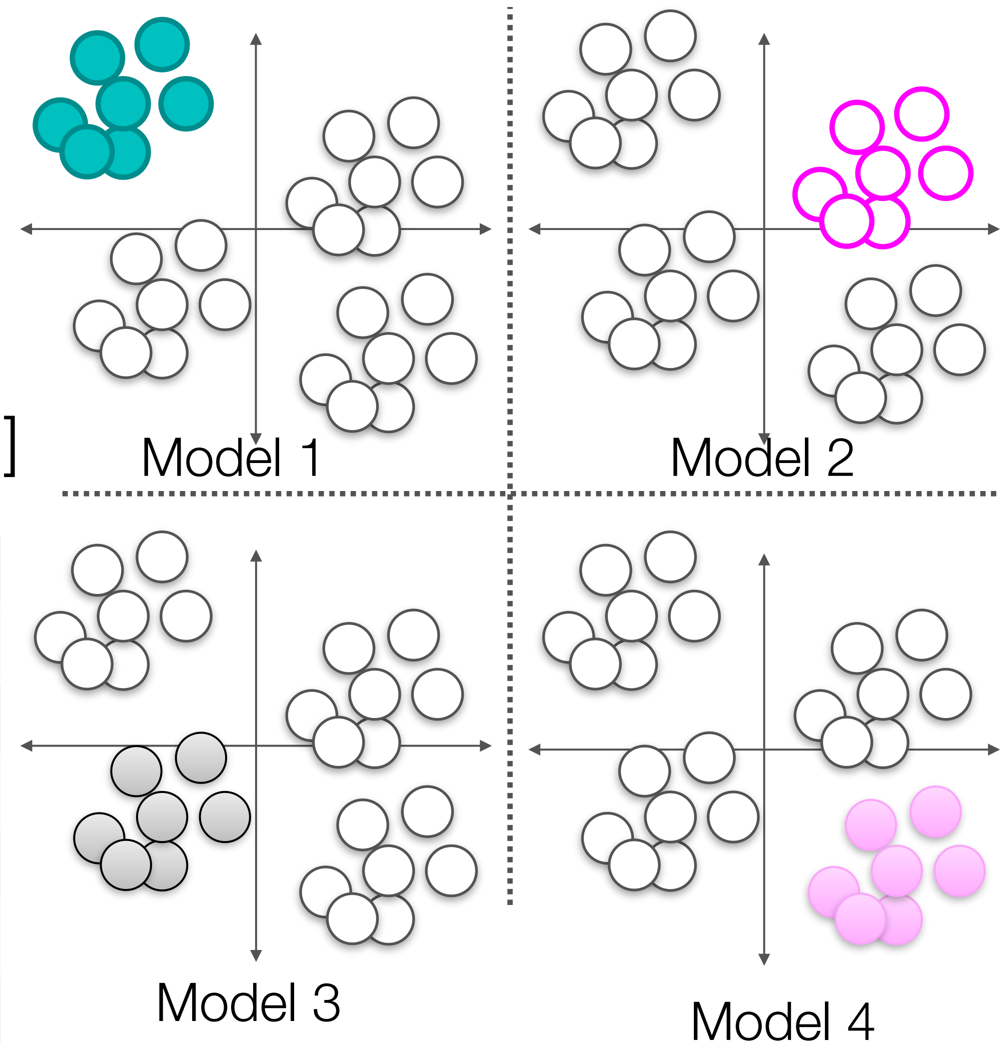
model probabilities

test data (rows)

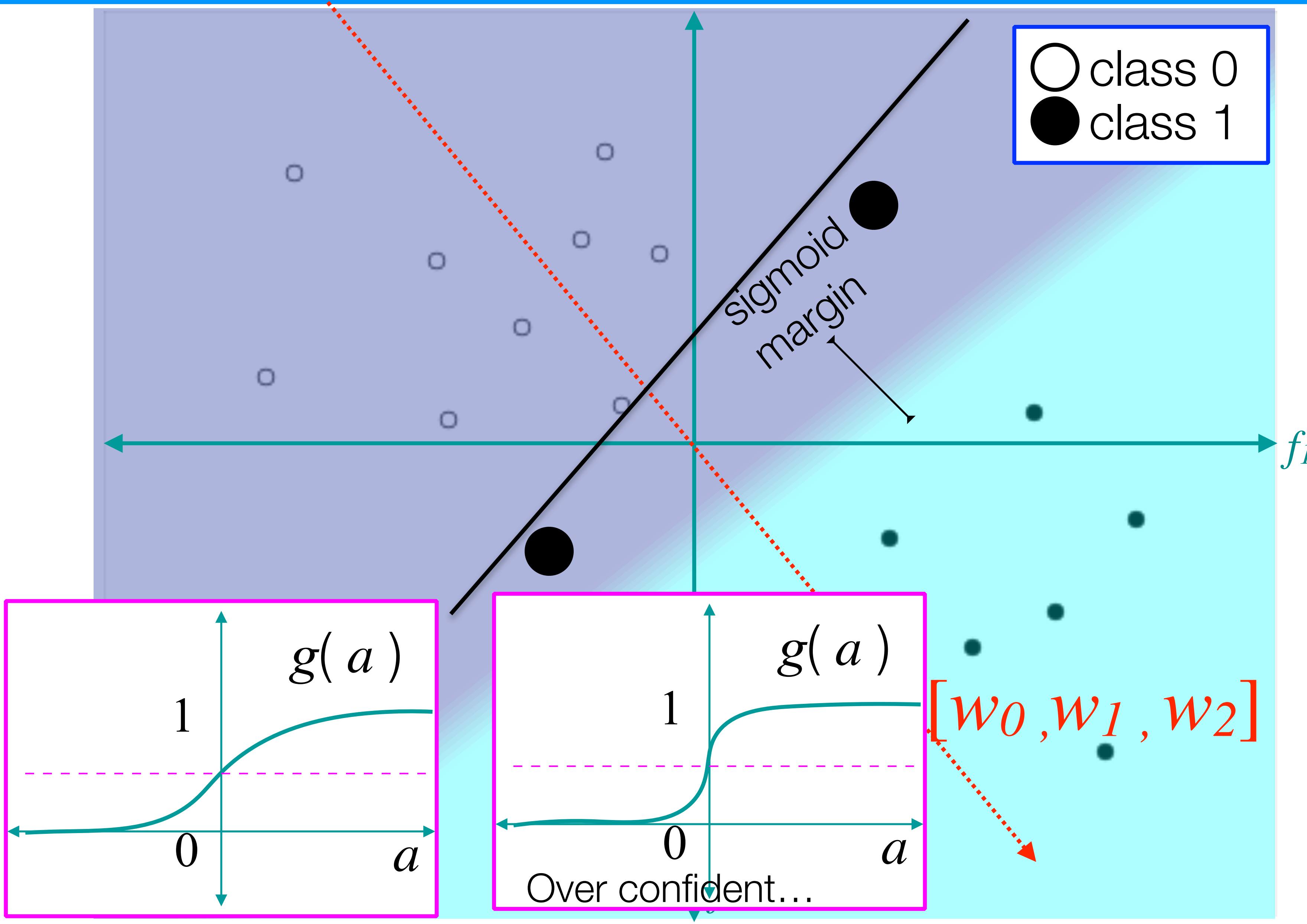
M1	M2	M3	M4	arg max	hash
0.91	*0.0*	*0.78*	*0.53*	*0*	
0.53	*0.5*	*0.32*	***0.91***	*3*	
0.01	*0.6*	***0.98***	*0.01*	*2*	
0.05	***0.83***	*0.21*	*0.81*	*1*	
0.85	*0.02*	*0.33*	*0.02*	*0*	
0.63	*0.11*	*0.09*	*0.33*	*0*	
0.01	*0.32*	*0.12*	***0.88***	*3*	



Binary formulations



Regularization, Intuition



`05. Logistic Regression.ipynb`

"Finish"

Programming

Vectorization

Regularization

Multi-class extension

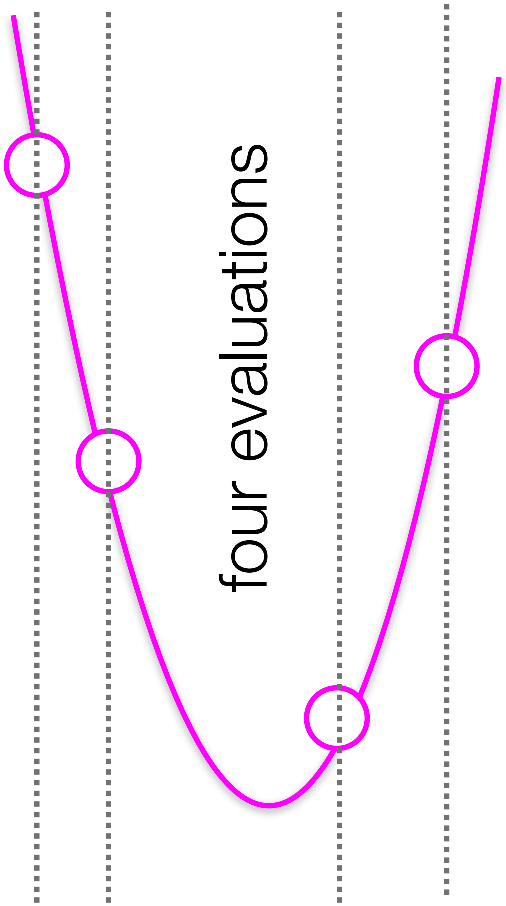
Demo Lecture

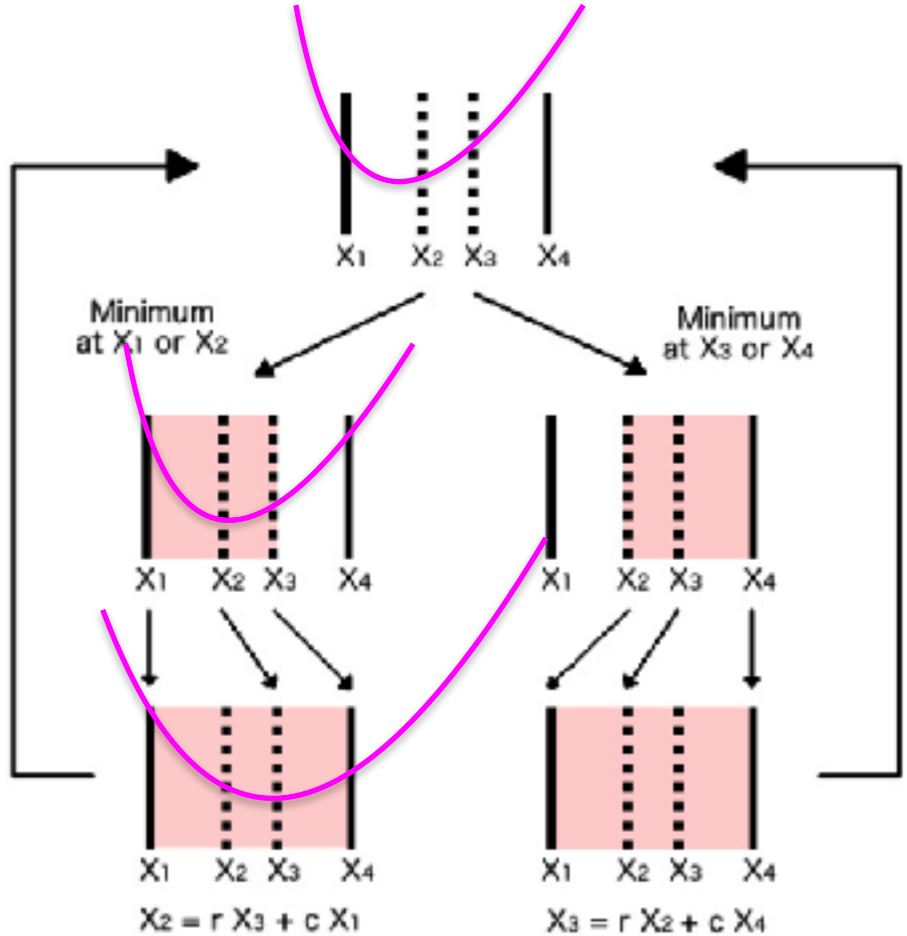
`06. Optimization`

Line Optimization

Intuition for the Hessian

Line Search: Intuition





How to optimally choose values to reduce required objective function evaluations?