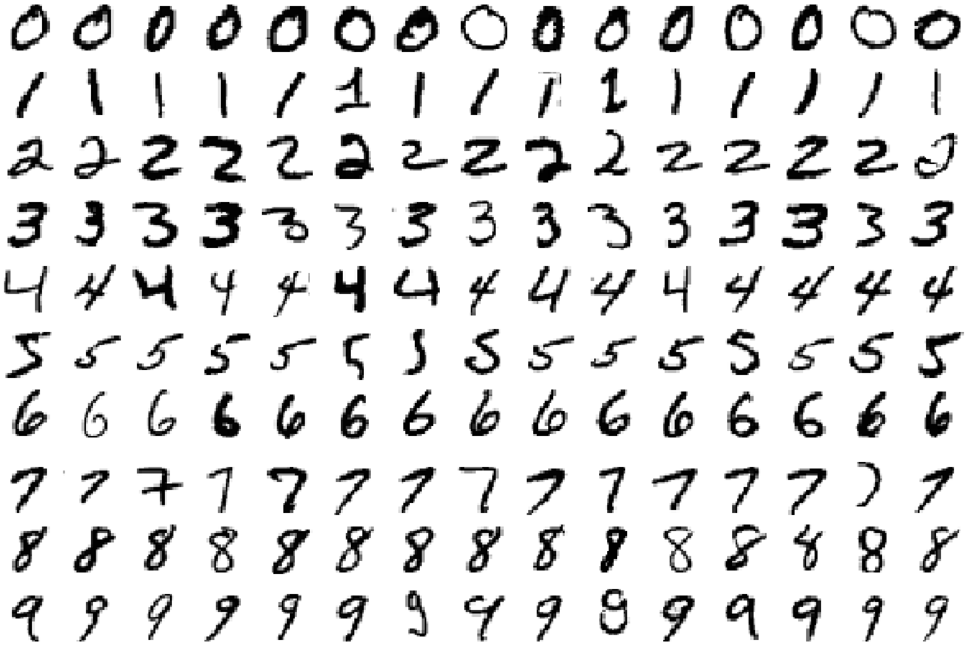
MNIST Classification: Multiple Linear Regression and Logistic Regression

CMPUT 328
Nilanjan Ray

MNIST Dataset



Classify images into digits

Each image is **28x28**

10 labels

55,000 training images

5,000 validation images

10,000 test images.

Linear regression

So, far we have seen:

Image (1x784)

Parameters (784x1)

$$\mathbf{y}^p = (\mathbf{x}-\bar{\mathbf{x}})\,\theta + \bar{y}$$

1 number, indicating digit

Mean vector of training images

Mean of training labels (digits)

[28x28]
Array of real numbers (784 numbers total)

Pixel values (feature) Digit

x_1	x_2	...	x_{784}	y
0.1	0.3	...	0.0	0
0.2	0.1	...	0.5	1
...	...	...	...	...
...	...	...	...	...
0.0	0.98	...	0.8	9
0.5	0.25	...	0.36	?
0.1	0.95	...	0.1	?

Multiple or Vector Linear Regression

Image (1x784)

Parameters (784x**10**)

$$\mathbf{y}^p=(\mathbf{x}-\bar{\mathbf{x}})W+\bar{\mathbf{y}}$$

[28x28]
Array of real numbers (784 numbers total)

Mean vector of training images

Mean of 1-hot training label vector

10 numbers, indicating class scores

Pixel values (feature) | Digit: 1-hot vector

x_1	x_2	...	x_{784}	y_1	...	y_{10}
0.1	0.3	...	0.0	0	...	1
0.2	0.1	...	0.5	1		0
...	...	...	...	...	...	...
...	...	...	...	...	...	...
0.0	0.98	...	0.8	0	...	1
0.5	0.25	...	0.36	?	...	?
0.1	0.95	...	0.1	?	...	?

Multiple Linear Regression: PyTorch Implementation

Prediction model: $\mathbf{y}^p = (\mathbf{x} - \bar{\mathbf{x}})W + \bar{\mathbf{y}}$

Regularized loss function: $L = \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{y}_i^p - \mathbf{y}_i\|^2 + \frac{\gamma}{2}\|W\|^2$

https://en.wikipedia.org/wiki/Matrix_calculus

This derivation requires matrix-vector differentiation

Gradient of loss function: $\boxed{\nabla L = (X^T X + \gamma I)W - X^T Y}$

where matrix X is defined as: $X = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} \\ \vdots \\ \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix}$

and matrix Y is defined as: $Y = \begin{bmatrix} \mathbf{y}_1 - \bar{\mathbf{y}} \\ \vdots \\ \mathbf{y}_n - \bar{\mathbf{y}} \end{bmatrix}$

and I is an identity matrix of size *784*-by-*784*

Equating gradient of L to zero matrix and solving for W gives us:

$$W = (X^T X + \gamma I)^{-1} X^T Y$$

We will “minimally” modify our linear regression scripts into multiple linear regression implementations!

Logistic Regression

Would it not be nice if we can predict class probabilities instead of scores?

image parameters

$$\mathbf{y}^p = f(\mathbf{x}, \mathbf{W})$$

prediction function
For logistic regression

10 numbers, indicating class probabilities

[28x28]
Array of real numbers
(784 numbers total)

Pixel values (feature)				Digit: 1-hot vector		
x_1	x_2	...	x_{784}	y_1	...	y_{10}
0.1	0.3	...	0.0	0	...	1
0.2	0.1	...	0.5	1		0
...	...	...	...	...	...	...
...	...	...	...	...	...	...
0.0	0.98	...	0.8	0	...	1
0.5	0.25	...	0.36	?	...	?
0.1	0.95	...	0.1	?	...	?

Logistic Regression

Can we modify scores from multiple regression function to output probabilities?

What is a suitable loss function for classification?

Logistic regression: from multiple linear regression

Scores from multiple linear regression: $\mathbf{s}_i = (\mathbf{x}_i - \bar{\mathbf{x}})W + \bar{\mathbf{y}}$ or $\mathbf{s}_{i,k} = (\mathbf{x}_i - \bar{\mathbf{x}})W_{:,k} + \bar{\mathbf{y}}_k$

Score for k^{th} class, $k = 0,\ldots,9$

Predicted probability for k^{th} class: $$y_{i,k}^{p} = \frac{\exp(s_{i,k})}{\sum_{c=0}^{9} \exp(s_{i,c})}$$

"Softmax" function

Logistic regression: loss function

Cross entropy loss: $loss(\mathbf{y}^p, \mathbf{y}) = -\sum_{k=0}^{9} \mathbf{y}_k \log(\boldsymbol{y}_k^p)$

Why this loss function? What does it mean? Why not use Euclidean loss as in MLR?