# 1 2019-01-15-applications

Let $z_1, \dots, z_n \in \mathbb{Z}_+$ be a set of $n$ non-negative integers, i.e. $\mathbb{Z}_+ = \{0, 1, 2, \dots\}$. Define for any positive $\mu > 0$ the function

$$f(\mu) = \sum_{i=1}^{n} \mu - z_i \log \mu,$$

where $\sum_{i=1}^{n}$ denotes the sum of $n$ terms, and log is the natural logarithm (log = ln).

1. Derive an expression in terms of $\mu, n, z_i$ for the first derivative $f'(\mu)$.

2. Derive an expression in terms of $n, z_i$ for the value of $\mu$ which is a critical point of $f$. Hint: set $f'(\mu) = 0$ and solve for $\mu$.

3. If $n = 3$ and $z_1 = 1, z_2 = 0, z_3 = 2$, then compute the value of the critical point $\mu =$ ______ and the critical function value $f(\mu) =$ ______. Fill in the blanks here but show your work below.

4. Is the critical point a minimum ($f''(\mu) > 0$), maximum ($f''(\mu) < 0$), or inflection point ($f''(\mu) = 0$)?

5. Write a function in the C programming language that computes the critical point $\mu$. The function should have two input arguments: the number $n$ of data `int n` and a pointer `int* z` to the data $z_i$. The function should output the critical point $\mu$ as a `double`.

## 2 2019-01-17-nearest-neighbors

Here are $n = 5$ data with $p = 2$ input dimensions. Each row is a person for which we have measured the height (first column of $X$, in centimeters), weight (second column in pounds). The output $y$ that we want to predict is diabetes diagnosis status (1=diabetes, 0=not).

$$X = \begin{bmatrix} 170 & 140 \\ 200 & 160 \\ 180 & 200 \\ 140 & 150 \\ 150 & 130 \end{bmatrix}, \ y = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \ d = \begin{bmatrix} \_\_\_\_\_ \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ \_\_\_\_\_ \end{bmatrix}$$

The goal is to compute the class predicted by the $K = 3$ nearest neighbor model, for a new/test person with features $x$=[height = 160 cm, weight = 130 pounds].

Assume that we use the Manhattan/L1 distance metric,

$$d(x, x') = \sum_{j=1}^{p} |x_j - x'_j|,$$

i.e. the total distance between $x, x'$ is the sum of distances on each of the two component dimensions (height and weight).

1. Fill in the blanks in the vector of distances $d$ above (each row should be the Manhattan/L1 distance between the new/test person and the training data).
2. Which are the three nearest neighbors? (write a star* for each of the three nearest neighbors in the blank next to the corresponding distances/rows)
3. What is the overall predicted class? (0 or 1)

## 3 2019-01-22-pseudocode

Below I have written pseudo-code for a version of the $k$-nearest neighbors algorithm. Fill in the blank on line 12 so that the algorithm computes predictions $\hat{y}_k$ for all $k \in \{1, \dots, K_{\max}\}$.

1: Function PRED1TOKMAXNEARESTNEIGHBORS<br>
2: Inputs: train inputs/features $x_1, \dots, x_n$, outputs/labels $y_1, \dots, y_n$,<br>
3:     test input/feature $x'$, max number of neighbors $K_{\max}$:<br>
4: **for** $i = 1$ to $n$ **do**:<br>
5:     $d_i \leftarrow \textsc{Distance}(x', x_i)$<br>
6: **end for**<br>
7: $t_1, \dots, t_n \leftarrow \textsc{SortedIndices}(d_1, \dots, d_n)$<br>
8: totalY $\leftarrow 0.0$<br>
9: **for** $k = 1$ to $K_{\max}$ **do**:<br>
10:     $i \leftarrow t_k$<br>
11:     totalY `+=` $y_i$<br>
12:     $\hat{y}_k \leftarrow$ _______________<br>
13: **end for**<br>
14: Output: predictions $\hat{y}_1, \dots, \hat{y}_{K_{\max}}$.

# 4 2019-01-24-cross-validation

The image below represents a training data set with $n = 70$ observations, one for each individual image of a digit. In order to perform cross-validation, fold ID numbers $\in \{1, 2, 3\}$ have been assigned to all observations/images in the corresponding row/letter.

A, fold=1

B, fold=2

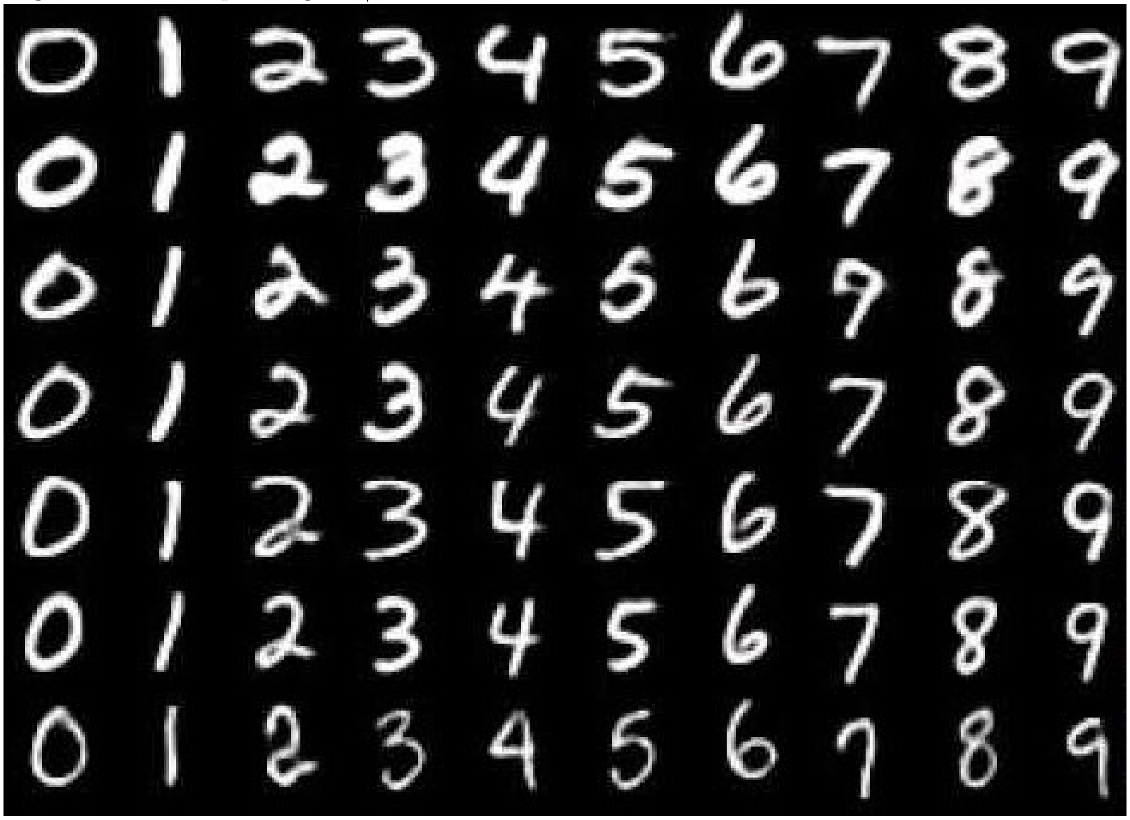
C, fold=2

D, fold=1

E, fold=3

F, fold=3

G, fold=3



1. For fold/split 1 which observations/letters are used for the training set? ____________________
   Which observations/letters are used for validation set? ____________________

2. For fold/split 2. Training set = ____________________, Validation set = ____________________.

3. For fold/split 3. Training set = ____________________, Validation set = ____________________.

# 5 2019-01-29-nearest-neighbors-code

Here are $n = 4$ data with $p = 2$ input dimensions. Each row is a person for which we have measured the height (first column of $X$, in centimeters), weight (second column in pounds). The output $y$ that we want to predict is a blood pressure measurement.

$$X = \begin{bmatrix} 170 & 140 \\ 200 & 160 \\ 180 & 200 \\ 140 & 150 \end{bmatrix}, \ y = \begin{bmatrix} 120 \\ 115 \\ 135 \\ 140 \end{bmatrix}$$

1. How would you represent these data in R? (fill in the blanks)

```
X <- matrix(c(_____,_____,_____,_____,_____,_____,_____,_____),
nrow=_____, ncol=_____)
y <- c(_____,_____,_____,_____)
```

2. Using `.C("myPrint_interface", as.double(X), PACKAGE="myPkg")` we can access the inputs via a C++ function:

```
void myPrint_interface(double *X_ptr){
  ...
}
```

Inside that function, what is the value of `X_ptr[4]`? _____________

## 6 2019-01-31-coding

Here is a block of C++ code which declares some variables that will be used for computing the nearest neighbors predictions for a multi-class classification problem with `n_labels=10` classes. Assume there are `nrow=1000` training observations in a feature/input space of size `ncol=256`. For each line of code, indicate (1) the total number of elements stored in the corresponding C array, (2) the underlying C type of each of those elements, double or int, and (3) YES if that line of code performs a dynamic memory allocation to get a new C array, otherwise NO.

```
Eigen::Map< Eigen::MatrixXd > train_inputs_mat(train_inputs_ptr, nrow, ncol);
```

(1)____________________ (2)____________________ (3)____________________

```
Eigen::Map< Eigen::VectorXd > test_input_vec(test_input_ptr, ncol);
```

(1)____________________ (2)____________________ (3)____________________

```
Eigen::VectorXd distance_vec(nrow);
```

(1)____________________ (2)____________________ (3)____________________

```
Eigen::VectorXi sorted_index_vec(nrow);
```

(1)____________________ (2)____________________ (3)____________________

```
Eigen::VectorXi label_count_vec(n_labels);
```

(1)____________________ (2)____________________ (3)____________________

# 7 2019-02-05-linear-regression

Let $w \in \mathbb{R}$ and $g(w) = \frac{1}{2}(w-4)^2$ be a cost function that we will minimize via gradient descent.

Derive an expression for gradient $\nabla g(w)$ in terms of $w$.

If we start at the origin $w^{(0)} = 0$, what is the starting value of the cost function? $g(w^{(0)}) =$ ____________

Let's use gradient descent with step size $\alpha = 0.5$ to find a lower cost. Recall that the gradient descent updates are given by $w^{(t)} = w^{(t-1)} - \alpha \nabla g(w^{(t-1)})$ for all $t \in \{1, 2, \ldots, \}$. What are the first two steps in gradient descent? Fill in the blanks below. (each should be a real number)

$\nabla g(w^{(0)}) =$ ____________

$w^{(1)} =$ ____________

$\nabla g(w^{(1)}) =$ ____________

$w^{(2)} =$ ____________

What is the ending value of the cost function? $g(w^{(2)}) =$ ____________

## 8 2019-02-07-logistic-regression

**Poisson regression** is a machine learning problem where the output/label $y_i \in \{0, 1, \dots\}$ is integer-valued, and the input/features $x_i \in \mathbb{R}^p$ is a real vector as usual. For example $y_i$ could be the number of pennies in your wallet, the number of cars in your garage, or the number of books in your backpack — all of these are non-negative integers.

This case needs special treatment because if you use standard linear regression, with the square loss, you end up with a prediction function $f(x_i) \in \mathbb{R}$ that predicts real numbers, and it does not make sense to predict a negative number (or a non-integer number) of pennies/cars/books. We will derive a loss function to use in this case.

We assume $y_i \sim \text{Poisson}(\lambda_i)$ where $\lambda_i \in \mathbb{R}_+$ is a non-negative real number — it is called the mean or rate parameter. The Poisson probability mass function is

$$\Pr(y_i, \lambda_i) = \frac{\lambda_i^{y_i} e^{-\lambda_i}}{y_i!}$$

Derive an expression in terms of $y_i$ and $\lambda_i$ for the log-likelihood of the mean parameter $\lambda_i$ given a single label $y_i$:

$\log \Pr(y_i, \lambda_i) =$ ________________________________________

We learn a linear function $f(x_i) = w^T x_i = \log \lambda_i \in \mathbb{R}$, which means that $\lambda_i = e^{w^T x_i}$.

The negative log-likelihood of a particular weight vector $w \in \mathbb{R}^p$ is therefore

$$\begin{aligned} -\text{LogLik}(w) &= -\sum_{i=1}^n \log \Pr(y_i, e^{w^T x_i}) \\ &= \sum_{i=1}^n \underline{\hspace{10cm}} \end{aligned}$$

In the blank above, write an expression for the negative log-likelihood in terms of $y_i, x_i, w$. There should be three terms that are added/subtracted together. Circle the term that does NOT depend on $w$ — the other two terms can be used as a loss function to minimize.

# 9 2019-02-12-log-reg-gradient

**Logistic regression** is a machine learning problem where the output/label $\tilde{y}_i \in \{-1, 1\}$ is binary-valued, and the inputs/features $x_i \in \mathbb{R}^p$ is a real vector as usual.

Let $X \in \mathbb{R}^{n \times p}$ be the input/feature matrix, and let $\tilde{y} \in \{-1, 1\}^n$ be the vector of labels. Let

$$\tilde{Y} = \text{Diag}(\tilde{y}) = \begin{bmatrix} \tilde{y}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \tilde{y}_n \end{bmatrix} \in \mathbb{R}^{n \times n}$$

be a matrix with labels on the diagonal and zeros elsewhere.

The total logistic loss for the linear prediction function $f(x_i) = w^T x_i$ is

$$\mathcal{L}(w) = \sum_{i=1}^{n} \log[1 + \exp(-\tilde{y}_i w^T x_i)].$$

Let

$$v = \begin{bmatrix} \frac{1}{1+\exp(\tilde{y}_1 w^T x_1)} \\ \vdots \\ \frac{1}{1+\exp(\tilde{y}_n w^T x_n)} \end{bmatrix} \in \mathbb{R}^n.$$

Derive an expression in terms of $X, \tilde{Y}, v$ for the gradient of the total logistic loss and put it in the blank below.

$\nabla \mathcal{L}(w) =$ ____________________________

## 10 2019-02-14-L2-regularization

In the statistics literature, the ridge regression problem is typically defined as follows. The output/label $y_i \in \mathbb{R}$ is real-valued, and the inputs/features $x_i \in \mathbb{R}^p$ is a real vector as usual. Let $X \in \mathbb{R}^{n \times p}$ be the input/feature matrix, and let $y \in \mathbb{R}^n$ be the vector of labels.

The linear prediction function is $f_{\beta,w}(x_i) = \beta + w^T x_i$, where $\beta \in \mathbb{R}$ is called the "intercept" or "bias" term, and $w \in \mathbb{R}^p$ is the usual vector of weights, one for each feature.

Let $1_n = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T \in \mathbb{R}^n$ be an $n$-vector of ones. The ridge regression cost function can then be defined as

$$\mathcal{C}_\lambda(\beta, w) = ||1_n\beta + Xw - y||_2^2 + \lambda||w||_2^2.$$

Note in the definition above that L2 regularization is only used for the weight vector $w$ (not for the bias/intercept $\beta$).

The optimal model parameters for a particular $\lambda \geq 0$ are defined as

$$\hat{\beta}^\lambda, \hat{w}^\lambda = \operatorname*{arg\,min}_{\beta \in \mathbb{R}, w \in \mathbb{R}^p} \mathcal{C}_\lambda(\beta, w).$$

To find the optimal model parameters we must first compute the gradients, (fill in the blanks below in terms of $X, y, w, \beta, 1_n$)

$\nabla_\beta \mathcal{C}_\lambda(\beta, w) =$ ____________________________

$\nabla_w \mathcal{C}_\lambda(\beta, w) =$ ____________________________

## 11 2019-02-26-line-search

Exact line search in 2 dimensions. For $w \in \mathbb{R}^2$, define the cost function

$$C(w) = \frac{1}{2}(w_1 - 1)^2 + \frac{1}{2}(w_2 + 1)^2 = \frac{1}{2}||w + \begin{bmatrix} -1 \\ 1 \end{bmatrix}||_2^2$$

Derive an expression for the gradient in terms of $w$, $\nabla C(w) =$____________________________
Let $w^{(0)} = 0$ be the starting point of gradient descent, at the origin.

The descent direction is
$d^{(0)} = -\nabla C(w^{(0)}) =$____________________________
The cost of a step with size $\alpha > 0$ in that direction is

$$\mathcal{C}_0(\alpha) = C(w^{(0)} + \alpha d^{(0)}).$$

To find the step size with the lowest cost we first need the derivative (in terms of $\alpha$):

$$\mathcal{C}_0'(\alpha) = \underline{\hspace{8cm}}$$

Setting the derivative to zero, $\mathcal{C}_0'(\alpha) = 0$, then solving
for $\alpha$ implies an optimal step size of $\alpha^{(0)} = \arg\min_\alpha \mathcal{C}_0(\alpha) =$____________________________

Taking that step lands us at

$$w^{(1)} = w^{(0)} + \alpha^{(0)} d^{(0)} = \underline{\hspace{8cm}}$$

which has a cost of

$$\mathcal{C}_0(\alpha^{(0)}) = C(w^{(1)}) = \underline{\hspace{8cm}}$$

## 12 2019-02-28-backtracking-line-search

**Exact line search for un-regularized least squares linear regression.**

For an input/feature matrix $X \in \mathbb{R}^{n \times p}$, an output/label vector $y \in \mathbb{R}^n$, and a weight vector $w \in \mathbb{R}^p$, define the least squares loss function

$$L(w) = \frac{1}{2}||Xw - y||_2^2$$

Derive an expression for the gradient in terms of $X, y, w$. $\nabla L(w) =$______________________________

The descent direction at $w$ is the negative gradient, $d = -\nabla L(w) =$______________________________

The cost of a step with size $\alpha > 0$ in that direction is

$$\mathcal{L}(\alpha) = L(w + \alpha d) = \underline{\hspace{6cm}}$$

To find the step size with the min cost we first need the derivative (in terms of $\alpha, d, w, X, y$):

$$\mathcal{L}'(\alpha) = \underline{\hspace{6cm}}$$

Setting the derivative to zero, $\mathcal{L}'(\alpha) = 0$, implies an optimal step size of

$\arg\min_\alpha \mathcal{L}'(\alpha) =$______________________________