# DAT200

Sammendrag av klassifikasjonsalgoritmer.

## Perceptron

1. Initialize the weights (close to zero random numbers)
2. For each training sample, $\mathbf{x}^{(i)}$:
    a. Compute the output value $\hat{y}$
    b. Update the weights.

```
w = close to zero random numbers
for i in iterations:
    for x_i in X:
        y_pred = dot(X, w) # -1 for < 0 and +1 for > 0
        dw_i = eta(y - y_pred)x_i
        w = [w_i + dw_i for w_i in w]
```

## Adaline

1. Initialize the weights (close to zero random numbers)
2. For each training sample, $\mathbf{x}^{(i)}$:
    a. Compute the output value $\phi(z)$
    b. Calculate the gradient descent.
    c. Update the weights.

```
w = close to zero random numbers
for i in iterations:
    y_pred = dot(X, w)
    for x_j in X.T: # Selects each column of X
        dw_j = eta(y - y_pred)x_j
        w_j = w_j + dw_j
```

## Logistic regression

1. Initialize the weights (close to zero random numbers)
2. For each training sample, $\mathbf{x}^{(i)}$:
    a. Compute the output value $\phi(z)$
    b. Calculate the gradient descent.
    c. Update the weights.

```
w = close to zero random numbers
for i in iterations:
    activation = dot(X, w)
    y_pred = 1 / (1 + np.exp(-activation))
    for x_j in X.T: # Selects each column of X
        dw_j = eta(y - y_pred)x_j
        w_j = w_j + dw_j
```

## Regularisation

Used alongside logistic regression (for example). Adds an extra term to the cost function and therefore the gradient descent.
The cost function for L2-regularisation is

$$J(\mathbf{w}) = -\sum_{i=1}^{n}\Big[-y^{(i)}log[\phi(z^{(i)})] - (1-y^{(i)})log[1-\phi(z^{(i)})]\Big] + \frac{\lambda}{2}||\mathbf{w}||^2$$