Foundations of Machine Learning

Linear Regression and Perceptron

Mahesan Niranjan

School of Electronics and Computer Science
University of Southampton

July 2021

Linear Regression & Perceptron

- Data: $\{\boldsymbol{x}_n, \ f_n\}_{n=1}^{N}$
  Input: $\boldsymbol{x}_n \in \mathcal{R}^p$; target / output f_n real valued
- Model: $f \ = \ \boldsymbol{w}^t \, \boldsymbol{x} \ + \ w_0$
  Output linear function of input (including a constant w_0)
- Work in $(p+1)$ dimensional space to avoid treating w_0 separately

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \quad \boldsymbol{a} = \begin{pmatrix} \boldsymbol{w} \\ w_0 \end{pmatrix}$$

- Data: $\{\boldsymbol{y}_n, \ f_n\}_{n=1}^{N}$
- Model: $f \ = \ \boldsymbol{y}^t \, \boldsymbol{a}$
- $p+1$ unknowns held in vector $\boldsymbol{a}$

Linear Regression & Perceptron

- Data: $\{\boldsymbol{x}_n, \; f_n\}_{n=1}^{N}$
  Input: $\boldsymbol{x}_n \in \mathcal{R}^p$; target / output f_n real valued
- Model: $f \;=\; \boldsymbol{w}^t \, \boldsymbol{x} \,+\, w_0$
  Output linear function of input (including a constant w_0)
- Work in $(p+1)$ dimensional space to avoid treating w_0 separately

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \quad \boldsymbol{a} = \begin{pmatrix} \boldsymbol{w} \\ w_0 \end{pmatrix}$$

- Data: $\{\boldsymbol{y}_n, \; f_n\}_{n=1}^{N}$
- Model: $f \;=\; \boldsymbol{y}^t \, \boldsymbol{a}$
- $p+1$ unknowns held in vector $\boldsymbol{a}$

Linear Regression & Perceptron

- Data: $\{\boldsymbol{x}_n, \ f_n\}_{n=1}^N$
  Input: $\boldsymbol{x}_n \in \mathcal{R}^p$; target / output f_n real valued
- Model: $f \ = \ \boldsymbol{w}^t \, \boldsymbol{x} \, + \, w_0$
  Output linear function of input (including a constant w_0)
- Work in $(p+1)$ dimensional space to avoid treating w_0 separately

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \quad \boldsymbol{a} = \begin{pmatrix} \boldsymbol{w} \\ w_0 \end{pmatrix}$$

- Data: $\{\boldsymbol{y}_n, \ f_n\}_{n=1}^N$
- Model: $f \ = \ \boldsymbol{y}^t \, \boldsymbol{a}$
- $p+1$ unknowns held in vector $\boldsymbol{a}$

Linear Regression & Perceptron

- Data: $\{\boldsymbol{x}_n,\ f_n\}_{n=1}^{N}$
  Input: $\boldsymbol{x}_n \in \mathcal{R}^p$; target / output f_n real valued
- Model: $f \ = \ \boldsymbol{w}^t\,\boldsymbol{x} \ + \ w_0$
  Output linear function of input (including a constant w_0)
- Work in $(p+1)$ dimensional space to avoid treating w_0 separately

$$\boldsymbol{y} \ = \ \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \quad \boldsymbol{a} \ = \ \begin{pmatrix} \boldsymbol{w} \\ w_0 \end{pmatrix}$$

- Data: $\{\boldsymbol{y}_n,\ f_n\}_{n=1}^{N}$
- Model: $f \ = \ \boldsymbol{y}^t\,\boldsymbol{a}$
- $p+1$ unknowns held in vector $\boldsymbol{a}$

Linear Regression & Perceptron

- Data: $\{\boldsymbol{x}_n, \; f_n\}_{n=1}^{N}$
  Input: $\boldsymbol{x}_n \in \mathcal{R}^p$; target / output f_n real valued
- Model: $f \;=\; \boldsymbol{w}^t \, \boldsymbol{x} \;+\; w_0$
  Output linear function of input (including a constant w_0)
- Work in $(p+1)$ dimensional space to avoid treating w_0 separately

$$\boldsymbol{y} \;=\; \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \quad \boldsymbol{a} \;=\; \begin{pmatrix} \boldsymbol{w} \\ w_0 \end{pmatrix}$$

- Data: $\{\boldsymbol{y}_n, \; f_n\}_{n=1}^{N}$
- Model: $f \;=\; \boldsymbol{y}^t \, \boldsymbol{a}$
- $p+1$ unknowns held in vector $\boldsymbol{a}$

Linear Regression & Perceptron

- Data: $\{\boldsymbol{x}_n,\ f_n\}_{n=1}^{N}$
  Input: $\boldsymbol{x}_n \in \mathcal{R}^p$; target / output f_n real valued
- Model: $f \ = \ \boldsymbol{w}^t\,\boldsymbol{x} \ + \ w_0$
  Output linear function of input (including a constant w_0)
- Work in $(p+1)$ dimensional space to avoid treating w_0 separately

$$\boldsymbol{y} \ = \ \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \quad \boldsymbol{a} \ = \ \begin{pmatrix} \boldsymbol{w} \\ w_0 \end{pmatrix}$$

- Data: $\{\boldsymbol{y}_n,\ f_n\}_{n=1}^{N}$
- Model: $f \ = \ \boldsymbol{y}^t\,\boldsymbol{a}$
- $p+1$ unknowns held in vector $\boldsymbol{a}$

Error and Minimization

- $E = \sum_{n=1}^{N} \left\{ \boldsymbol{y}_n^t \boldsymbol{a} - f_n \right\}^2$
- $E = \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\}^2$
- To find the best $\boldsymbol{a}$ we minimize E – differentiate with respect to each of the unknowns in $\boldsymbol{a}$ and set to zero.
- $$\frac{\partial E}{\partial a_i} = 2 \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\} (y_{ni})$$
- There are $(p+1)$ derivatives (with respect to each a_i)
- Equating them to zero gives $(p+1)$ equations in $(p+1)$ unknowns

Error and Minimization

- $E = \sum_{n=1}^{N} \left\{ \boldsymbol{y}_n^t \boldsymbol{a} - f_n \right\}^2$
- $E = \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\}^2$
- To find the best $\boldsymbol{a}$ we minimize E – differentiate with respect to each of the unknowns in $\boldsymbol{a}$ and set to zero.
- $$\frac{\partial E}{\partial a_i} = 2 \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\} (y_{ni})$$
- There are $(p+1)$ derivatives (with respect to each a_i)
- Equating them to zero gives $(p+1)$ equations in $(p+1)$ unknowns

Error and Minimization

- $E = \sum_{n=1}^{N} \left\{ \boldsymbol{y}_n^t \boldsymbol{a} - f_n \right\}^2$
- $E = \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\}^2$
- To find the best $\boldsymbol{a}$ we minimize E – differentiate with respect to each of the unknowns in $\boldsymbol{a}$ and set to zero.
- $$\frac{\partial E}{\partial a_i} = 2 \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\} (y_{ni})$$
- There are $(p+1)$ derivatives (with respect to each a_i)
- Equating them to zero gives $(p+1)$ equations in $(p+1)$ unknowns

Error and Minimization

- $E = \sum_{n=1}^{N} \left\{ \boldsymbol{y}_n^t \boldsymbol{a} - f_n \right\}^2$
- $E = \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\}^2$
- To find the best $\boldsymbol{a}$ we minimize E – differentiate with respect to each of the unknowns in $\boldsymbol{a}$ and set to zero.
- $$\frac{\partial E}{\partial a_i} = 2 \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\} (y_{ni})$$
- There are $(p+1)$ derivatives (with respect to each a_i)
- Equating them to zero gives $(p+1)$ equations in $(p+1)$ unknowns

Error and Minimization

- $E = \sum_{n=1}^{N} \left\{ \boldsymbol{y}_n^t \boldsymbol{a} - f_n \right\}^2$
- $E = \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\}^2$
- To find the best $\boldsymbol{a}$ we minimize E – differentiate with respect to each of the unknowns in $\boldsymbol{a}$ and set to zero.
-
$$\frac{\partial E}{\partial a_i} = 2 \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\} (y_{ni})$$
- There are $(p+1)$ derivatives (with respect to each a_i)
- Equating them to zero gives $(p+1)$ equations in $(p+1)$ unknowns

Error and Minimization

- $E = \sum_{n=1}^{N} \left\{ \boldsymbol{y}_n^t \boldsymbol{a} - f_n \right\}^2$
- $E = \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\}^2$
- To find the best $\boldsymbol{a}$ we minimize E – differentiate with respect to each of the unknowns in $\boldsymbol{a}$ and set to zero.
- $$\frac{\partial E}{\partial a_i} = 2 \sum_{n=1}^{N} \left\{ \left(\sum_{j=1}^{(p+1)} a_j \, y_{nj} \right) - f_n \right\} (y_{ni})$$
- There are $(p+1)$ derivatives (with respect to each a_i)
- Equating them to zero gives $(p+1)$ equations in $(p+1)$ unknowns

Solution to Regression

- $(p+1)$ simultaneous equations to solve:
  i^{th} row, j^{th} column shown

$$\begin{pmatrix} & \cdots & \\ \cdots & \cdots & \cdots \\ \vdots & \sum_{n=1}^{N} y_{ni} y_{nj} & \cdots \\ \vdots & \cdots & \vdots \\ \cdots & \cdots & \cdots \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{(p+1)} \end{pmatrix} = \begin{pmatrix} \vdots \\ \sum_{n=1}^{N} f_n y_{ni} \\ \vdots \end{pmatrix}$$

Derivation in vector/matrix form

- $\boldsymbol{Y}$: $N \times (p+1)$ matrix n^{th} row is $\boldsymbol{y}_n^t$
- $\boldsymbol{f}$: $N \times 1$ vector of outputs
- Error $E = ||\boldsymbol{Y}\,\boldsymbol{a} - \boldsymbol{f}||^2$
- Homework: Verify the error written like this is the same as the one we wrote out in lengthy algebra.
- Gradient

$$\nabla_{\boldsymbol{a}} E = 2\boldsymbol{Y}^t(\boldsymbol{Ya} - \boldsymbol{f})$$

- Equating the gradient to zero gives

$$\begin{aligned} \boldsymbol{Y}^t\boldsymbol{Ya} &= \boldsymbol{Y}^t\boldsymbol{f} \\ \boldsymbol{a} &= \left(\boldsymbol{Y}^t\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^t\boldsymbol{f} \end{aligned}$$

- Homework: With three data points in one dimensional input space (x_1, f_1), (x_2, f_2) and (x_3, f_3) and two unknowns, slope (m) and intercept (c) of fitting a straight line, write out all the expressions seen so far.

Derivation in vector/matrix form

- $\boldsymbol{Y}$: $N \times (p+1)$ matrix n^{th} row is $\boldsymbol{y}_n^t$
- $\boldsymbol{f}$: $N \times 1$ vector of outputs
- Error $E = ||\boldsymbol{Y}\,\boldsymbol{a} - \boldsymbol{f}||^2$
- $\boxed{\text{Homework:}}$ Verify the error written like this is the same as the one we wrote out in lengthy algebra.
- Gradient

$$\nabla_{\boldsymbol{a}} E = 2\boldsymbol{Y}^t\,(\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{f})$$

- Equating the gradient to zero gives

$$\begin{aligned} \boldsymbol{Y}^t\boldsymbol{Y}\boldsymbol{a} &= \boldsymbol{Y}^t\boldsymbol{f} \\ \boldsymbol{a} &= \left(\boldsymbol{Y}^t\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^t\boldsymbol{f} \end{aligned}$$

- $\boxed{\text{Homework:}}$ With three data points in one dimensional input space $(x_1,\ f_1)$, $(x_2,\ f_2)$ and $(x_3,\ f_3)$ and two unknowns, slope (m) and intercept (c) of fitting a straight line, write out all the expressions seen so far.

Derivation in vector/matrix form

- $\boldsymbol{Y}$: $N \times (p+1)$ matrix n^{th} row is $\boldsymbol{y}_n^t$
- $\boldsymbol{f}$: $N \times 1$ vector of outputs
- Error $E \ = \ ||\boldsymbol{Y}\,\boldsymbol{a} - \boldsymbol{f}||^2$
- $\boxed{\text{Homework:}}$ Verify the error written like this is the same as the one we wrote out in lengthy algebra.
- Gradient

$$\nabla_{\boldsymbol{a}} E \ = \ 2\boldsymbol{Y}^t\,(\boldsymbol{Y}\boldsymbol{a} \ - \ \boldsymbol{f})$$

- Equating the gradient to zero gives

$$\begin{aligned} \boldsymbol{Y}^t\boldsymbol{Y}\boldsymbol{a} &= \boldsymbol{Y}^t\boldsymbol{f} \\ \boldsymbol{a} &= \left(\boldsymbol{Y}^t\boldsymbol{Y}\right)^{-1}\,\boldsymbol{Y}^t\,\boldsymbol{f} \end{aligned}$$

- $\boxed{\text{Homework:}}$ With three data points in one dimensional input space $(x_1,\ f_1)$, $(x_2,\ f_2)$ and $(x_3,\ f_3)$ and two unknowns, slope (m) and intercept (c) of fitting a straight line, write out all the expressions seen so far.

Derivation in vector/matrix form

- $\boldsymbol{Y}$: $N \times (p+1)$ matrix n^{th} row is $\boldsymbol{y}_n^t$
- $\boldsymbol{f}$: $N \times 1$ vector of outputs
- Error $E \;=\; ||\boldsymbol{Y}\,\boldsymbol{a} - \boldsymbol{f}||^2$
- $\boxed{\text{Homework:}}$ Verify the error written like this is the same as the one we wrote out in lengthy algebra.
- Gradient

$$\nabla_{\boldsymbol{a}} E \;=\; 2\,\boldsymbol{Y}^t\,(\boldsymbol{Y}\boldsymbol{a} \;-\; \boldsymbol{f})$$

- Equating the gradient to zero gives

$$\begin{aligned} \boldsymbol{Y}^t\,\boldsymbol{Y}\boldsymbol{a} &= \boldsymbol{Y}^t\boldsymbol{f} \\ \boldsymbol{a} &= \left(\boldsymbol{Y}^t\boldsymbol{Y}\right)^{-1}\,\boldsymbol{Y}^t\,\boldsymbol{f} \end{aligned}$$

- $\boxed{\text{Homework:}}$ With three data points in one dimensional input space $(x_1,\ f_1)$, $(x_2,\ f_2)$ and $(x_3,\ f_3)$ and two unknowns, slope (m) and intercept (c) of fitting a straight line, write out all the expressions seen so far.

Derivation in vector/matrix form

- $\boldsymbol{Y}$: $N \times (p+1)$ matrix n^{th} row is $\boldsymbol{y}_n^t$
- $\boldsymbol{f}$: $N \times 1$ vector of outputs
- Error $E = ||\boldsymbol{Y}\,\boldsymbol{a} - \boldsymbol{f}||^2$
- $\boxed{\text{Homework:}}$ Verify the error written like this is the same as the one we wrote out in lengthy algebra.
- Gradient

$$\nabla_{\boldsymbol{a}} E = 2\boldsymbol{Y}^t\,(\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{f})$$

- Equating the gradient to zero gives

$$\begin{aligned} \boldsymbol{Y}^t\boldsymbol{Y}\boldsymbol{a} &= \boldsymbol{Y}^t\boldsymbol{f} \\ \boldsymbol{a} &= \left(\boldsymbol{Y}^t\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^t\boldsymbol{f} \end{aligned}$$

- $\boxed{\text{Homework:}}$ With three data points in one dimensional input space (x_1, f_1), (x_2, f_2) and (x_3, f_3) and two unknowns, slope (m) and intercept (c) of fitting a straight line, write out all the expressions seen so far.

Derivation in vector/matrix form

- $\boldsymbol{Y}$: $N \times (p+1)$ matrix n^{th} row is $\boldsymbol{y}_n^t$
- $\boldsymbol{f}$: $N \times 1$ vector of outputs
- Error $E = ||\boldsymbol{Y}\,\boldsymbol{a} - \boldsymbol{f}||^2$
- Homework: Verify the error written like this is the same as the one we wrote out in lengthy algebra.
- Gradient

$$\nabla_{\boldsymbol{a}} E = 2\boldsymbol{Y}^t\,(\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{f})$$

- Equating the gradient to zero gives

$$\begin{aligned}\boldsymbol{Y}^t\boldsymbol{Y}\boldsymbol{a} &= \boldsymbol{Y}^t\boldsymbol{f}\\ \boldsymbol{a} &= \left(\boldsymbol{Y}^t\boldsymbol{Y}\right)^{-1}\,\boldsymbol{Y}^t\,\boldsymbol{f}\end{aligned}$$

- Homework: With three data points in one dimensional input space $(x_1,\ f_1)$, $(x_2,\ f_2)$ and $(x_3,\ f_3)$ and two unknowns, slope (m) and intercept (c) of fitting a straight line, write out all the expressions seen so far.

Derivation in vector/matrix form

- $\boldsymbol{Y}$: $N \times (p+1)$ matrix n^{th} row is $\boldsymbol{y}_n^t$
- $\boldsymbol{f}$: $N \times 1$ vector of outputs
- Error $E \;=\; ||\boldsymbol{Y}\,\boldsymbol{a} - \boldsymbol{f}||^2$
- $\boxed{\text{Homework:}}$ Verify the error written like this is the same as the one we wrote out in lengthy algebra.
- Gradient

$$\nabla_{\boldsymbol{a}} E \;=\; 2\,\boldsymbol{Y}^t\,(\boldsymbol{Y}\boldsymbol{a} \;-\; \boldsymbol{f})$$

- Equating the gradient to zero gives

$$\begin{aligned} \boldsymbol{Y}^t\,\boldsymbol{Y}\boldsymbol{a} &= \boldsymbol{Y}^t\,\boldsymbol{f} \\ \boldsymbol{a} &= \left(\boldsymbol{Y}^t\,\boldsymbol{Y}\right)^{-1}\;\boldsymbol{Y}^t\,\boldsymbol{f} \end{aligned}$$

- $\boxed{\text{Homework:}}$ With three data points in one dimensional input space $(x_1,\ f_1)$, $(x_2,\ f_2)$ and $(x_3,\ f_3)$ and two unknowns, slope (m) and intercept (c) of fitting a straight line, write out all the expressions seen so far.

Solution by Gradient Descent

- Gradient vector: $\nabla_{\boldsymbol{a}} E = 2\boldsymbol{Y}^t(\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{f})$
- Steepest descent algorithm:

```
Initialize a at random
Update a^(k+1) = a^(k) − η ∇_a E
Until Convergence
```

$$\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} - \eta \nabla_{\boldsymbol{a}} E$$

- Second order (Newton's) method

```
Initialize a at random
Update a^(k+1) = a^(k) − H^-1 ∇_a E
Until Convergence
```

$$\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{a}} E$$

- Rapid convergence with second order method, but cost of computing and inverting $\boldsymbol{H}$ can be high (more on this under Neural Networks)

Solution by Gradient Descent

- Gradient vector: $\nabla_{\boldsymbol{a}} E = 2\boldsymbol{Y}^t(\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{f})$
- Steepest descent algorithm:

$$
\begin{array}{l}
\texttt{Initialize } \boldsymbol{a} \texttt{ at random} \\
\texttt{Update } \boldsymbol{a}^{(k+1)} = a^{(k)} - \eta \nabla_{\boldsymbol{a}} E \\
\texttt{Until Convergence}
\end{array}
$$

- Second order (Newton's) method

$$
\begin{array}{l}
\texttt{Initialize } \boldsymbol{a} \texttt{ at random} \\
\texttt{Update } \boldsymbol{a}^{(k+1)} = a^{(k)} - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{a}} E \\
\texttt{Until Convergence}
\end{array}
$$

- Rapid convergence with second order method, but cost of computing and inverting $\boldsymbol{H}$ can be high (more on this under Neural Networks)

Solution by Gradient Descent

- Gradient vector: $\nabla_{\boldsymbol{a}} E = 2\boldsymbol{Y}^t(\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{f})$
- Steepest descent algorithm:

  Initialize $\boldsymbol{a}$ at random
  Update $\boldsymbol{a}^{(k+1)} = a^{(k)} - \eta \nabla_{\boldsymbol{a}} E$
  Until Convergence

- Second order (Newton's) method

  Initialize $\boldsymbol{a}$ at random
  Update $\boldsymbol{a}^{(k+1)} = a^{(k)} - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{a}} E$
  Until Convergence

- Rapid convergence with second order method, but cost of computing and inverting $\boldsymbol{H}$ can be high (more on this under Neural Networks)

Solution by Gradient Descent

- Gradient vector: $\nabla_{\boldsymbol{a}} E = 2\boldsymbol{Y}^t(\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{f})$
- Steepest descent algorithm:

$$\begin{array}{l} \texttt{Initialize } \boldsymbol{a} \texttt{ at random} \\ \texttt{Update } \boldsymbol{a}^{(k+1)} = a^{(k)} - \eta \nabla_{\boldsymbol{a}} E \\ \texttt{Until Convergence} \end{array}$$

- Second order (Newton's) method

$$\begin{array}{l} \texttt{Initialize } \boldsymbol{a} \texttt{ at random} \\ \texttt{Update } \boldsymbol{a}^{(k+1)} = a^{(k)} - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{a}} E \\ \texttt{Until Convergence} \end{array}$$

- Rapid convergence with second order method, but cost of computing and inverting $\boldsymbol{H}$ can be high (more on this under Neural Networks)

Gradient and Stochastic Gradient Descent

- Error $E = \sum_{n=1}^{N} e_n^2$
- True gradient:

$$\nabla_{\boldsymbol{a}} E = 2 \sum_{n=1}^{N} \left\{ \mathbf{y}_n^t \mathbf{a} - \mathbf{f}_n \right\} (\mathbf{y}_n)$$

- Gradient computed on n^{th} data:

$$\nabla_{\boldsymbol{a}} e_n = 2 \left\{ \mathbf{y}_n^t \mathbf{a} - \mathbf{f}_n \right\} (\mathbf{y}_n)$$

Gradient and Stochastic Gradient Descent

- Error $E = \sum_{n=1}^{N} e_n^2$
- True gradient:

$$\nabla_{\boldsymbol{a}} E = 2 \sum_{n=1}^{N} \left\{ \mathbf{y}_\mathbf{n}^\mathbf{t} \mathbf{a} - \mathbf{f}_\mathbf{n} \right\} (\mathbf{y}_\mathbf{n})$$

- Gradient computed on n^{th} data:

$$\nabla_{\boldsymbol{a}} e_n = 2 \left\{ \mathbf{y}_\mathbf{n}^\mathbf{t} \mathbf{a} - \mathbf{f}_\mathbf{n} \right\} (\mathbf{y}_\mathbf{n})$$

Gradient and Stochastic Gradient Descent

- Error $E = \sum_{n=1}^{N} e_n^2$
- True gradient:

$$\nabla_{\boldsymbol{a}} E = 2 \sum_{n=1}^{N} \left\{ \mathbf{y_n^t a} - \mathbf{f_n} \right\} (\mathbf{y_n})$$

- Gradient computed on n^{th} data:

$$\nabla_{\boldsymbol{a}}\, e_n = 2 \left\{ \mathbf{y_n^t a} - \mathbf{f_n} \right\} (\mathbf{y_n})$$

Regularization

- Pseudo inverse solution: $\boldsymbol{a} = \left(\boldsymbol{Y}^t \boldsymbol{Y}\right)^{-1} \boldsymbol{Y}^t \boldsymbol{f}$
- This can be ill conditioned, so we could *regularize* by

$$\boldsymbol{a} = \left(\boldsymbol{Y}^t \boldsymbol{Y} + \gamma \boldsymbol{I}\right)^{-1} \boldsymbol{Y}^t \boldsymbol{f}$$

  where γ is a small constant.
- We achieve precisely this by minimizing an error of the form

$$||\boldsymbol{Y}\boldsymbol{a} - \boldsymbol{f}||^2 + \gamma||\boldsymbol{a}||^2$$

  Here a quadratic penalty term has been included
- $\boxed{\text{Homework:}}$ Differentiate this error and derive the regularized solution
- Sparse solutions are obtained by regularizing with an l_1 norm (sum of absolute values of $\boldsymbol{a}$, i.e. $\sum_{j=1}^{p} |a_j|$); See **Lab 4.**

Regularization

- Pseudo inverse solution: $\boldsymbol{a} = \left(\boldsymbol{Y}^t \boldsymbol{Y}\right)^{-1} \boldsymbol{Y}^t \boldsymbol{f}$
- This can be ill conditioned, so we could *regularize* by

$$\boldsymbol{a} = \left(\boldsymbol{Y}^t \boldsymbol{Y} + \gamma \boldsymbol{I}\right)^{-1} \boldsymbol{Y}^t \boldsymbol{f}$$

  where γ is a small constant.
- We achieve precisely this by minimizing an error of the form

$$||\boldsymbol{Ya} - \boldsymbol{f}||^2 + \gamma||\boldsymbol{a}||^2$$

  Here a quadratic penalty term has been included
- $\boxed{\text{Homework:}}$ Differentiate this error and derive the regularized solution
- Sparse solutions are obtained by regularizing with an l_1 norm (sum of absolute values of $\boldsymbol{a}$, i.e. $\sum_{j=1}^{p} |a_j|$); See **Lab 4**.

Regularization

- Pseudo inverse solution: $\boldsymbol{a} = \left(\boldsymbol{Y}^t \boldsymbol{Y}\right)^{-1} \boldsymbol{Y}^t \boldsymbol{f}$
- This can be ill conditioned, so we could *regularize* by

$$\boldsymbol{a} = \left(\boldsymbol{Y}^t \boldsymbol{Y} + \gamma \boldsymbol{I}\right)^{-1} \boldsymbol{Y}^t \boldsymbol{f}$$

  where γ is a small constant.
- We achieve precisely this by minimizing an error of the form

$$||\boldsymbol{Ya} - \boldsymbol{f}||^2 + \gamma||\boldsymbol{a}||^2$$

  Here a quadratic penalty term has been included
- Homework: Differentiate this error and derive the regularized solution
- Sparse solutions are obtained by regularizing with an l_1 norm (sum of absolute values of $\boldsymbol{a}$, i.e. $\sum_{j=1}^{p} |a_j|$); See **Lab 4**.

Regularization

- Pseudo inverse solution: $\boldsymbol{a} = \left(\boldsymbol{Y}^t \boldsymbol{Y}\right)^{-1} \boldsymbol{Y}^t \boldsymbol{f}$
- This can be ill conditioned, so we could *regularize* by

$$\boldsymbol{a} = \left(\boldsymbol{Y}^t \boldsymbol{Y} + \gamma \boldsymbol{I}\right)^{-1} \boldsymbol{Y}^t \boldsymbol{f}$$

  where γ is a small constant.
- We achieve precisely this by minimizing an error of the form

$$||\boldsymbol{Ya} - \boldsymbol{f}||^2 + \gamma ||\boldsymbol{a}||^2$$

  Here a quadratic penalty term has been included
- $\boxed{\text{Homework:}}$ Differentiate this error and derive the regularized solution
- Sparse solutions are obtained by regularizing with an l_1 norm (sum of absolute values of $\boldsymbol{a}$, i.e. $\sum_{j=1}^{p} |a_j|$); See **Lab 4**.

Regularization

- Pseudo inverse solution: $\boldsymbol{a} = \left(\boldsymbol{Y}^t\boldsymbol{Y}\right)^{-1} \boldsymbol{Y}^t\boldsymbol{f}$
- This can be ill conditioned, so we could *regularize* by

$$\boldsymbol{a} = \left(\boldsymbol{Y}^t\boldsymbol{Y} + \gamma\boldsymbol{I}\right)^{-1} \boldsymbol{Y}^t\boldsymbol{f}$$

  where γ is a small constant.
- We achieve precisely this by minimizing an error of the form

$$||\boldsymbol{Ya} - \boldsymbol{f}||^2 + \gamma||\boldsymbol{a}||^2$$

  Here a quadratic penalty term has been included
- $\boxed{\text{Homework:}}$ Differentiate this error and derive the regularized solution
- Sparse solutions are obtained by regularizing with an l_1 norm (sum of absolute values of $\boldsymbol{a}$, i.e. $\sum_{j=1}^{p}|a_j|$); See **Lab 4**.

Perceptron

A suitable performance measure

- Number of misclassified examples as measure of error
  Piecewise constant (cannot differentiate)
- Suitable error measure:

$$E_P = -\sum \boldsymbol{y}_n^t \boldsymbol{a}$$

  - Summation taken over misclassified examples
  - We started with $\boldsymbol{y}_n^t\boldsymbol{a} > 0$ for positive class and $\boldsymbol{y}_n^t\boldsymbol{a} < 0$ for the negative class; we then switch the signs of negative class examples and required $\boldsymbol{y}_n^t\boldsymbol{a} > 0$ for all the training data; so for the misclassified examples $-\sum \boldsymbol{y}_n^t \boldsymbol{a}$ should be as small as possible.

Perceptron

A suitable performance measure

- Number of misclassified examples as measure of error
  Piecewise constant (cannot differentiate)
- Suitable error measure:

$$E_P = -\sum \boldsymbol{y}_n^t \boldsymbol{a}$$

- Summation taken over misclassified examples
- We started with $\boldsymbol{y}_n^t \boldsymbol{a} > 0$ for positive class and $\boldsymbol{y}_n^t \boldsymbol{a} < 0$ for the negative class; we then switch the signs of negative class examples and required $\boldsymbol{y}_n^t \boldsymbol{a} > 0$ for all the training data; so for the misclassified examples $-\sum \boldsymbol{y}_n^t \boldsymbol{a}$ should be as small as possible.

Perceptron

A suitable performance measure

- Number of misclassified examples as measure of error
  Piecewise constant (cannot differentiate)
- Suitable error measure:

$$E_P = -\sum \boldsymbol{y}_n^t \boldsymbol{a}$$

  - Summation taken over misclassified examples
  - We started with $\boldsymbol{y}_n^t \boldsymbol{a} > 0$ for positive class and $\boldsymbol{y}_n^t \boldsymbol{a} < 0$ for the negative class; we then switch the signs of negative class examples and required $\boldsymbol{y}_n^t \boldsymbol{a} > 0$ for all the training data; so for the misclassified examples $-\sum \boldsymbol{y}_n^t \boldsymbol{a}$ should be as small as possible.

- Number of misclassified examples as measure of error
  Piecewise constant (cannot differentiate)
- Suitable error measure:

$$E_P = -\sum \boldsymbol{y}_n^t \boldsymbol{a}$$

  - Summation taken over misclassified examples
  - We started with $\boldsymbol{y}_n^t \boldsymbol{a} > 0$ for positive class and $\boldsymbol{y}_n^t \boldsymbol{a} < 0$ for the negative class; we then switch the signs of negative class examples and required $\boldsymbol{y}_n^t \boldsymbol{a} > 0$ for all the training data; so for the misclassified examples $-\sum \boldsymbol{y}_n^t \boldsymbol{a}$ should be as small as possible.

Perceptron

A suitable performance measure

- Number of misclassified examples as measure of error
  Piecewise constant (cannot differentiate)
- Suitable error measure:

$$E_P = -\sum \boldsymbol{y}_n^t \boldsymbol{a}$$

  - Summation taken over misclassified examples
  - We started with $\boldsymbol{y}_n^t \boldsymbol{a} > 0$ for positive class and $\boldsymbol{y}_n^t \boldsymbol{a} < 0$ for the negative class; we then switch the signs of negative class examples and required $\boldsymbol{y}_n^t \boldsymbol{a} > 0$ for all the training data; so for the misclassified examples $-\sum \boldsymbol{y}_n^t \boldsymbol{a}$ should be as small as possible.

- Gradient:

$$\frac{\partial E}{\partial \boldsymbol{a}} = -\sum \boldsymbol{y}_n$$

- Gradient algorithm: $\boldsymbol{a}^{(k+1)} = a^{(k)} + \sum \boldsymbol{y}_n$
- Stochastic gradient algorithm:

$$\boldsymbol{a}^{(k+1)} = a^{(k)} + \boldsymbol{y}_n$$

- Note what $\boldsymbol{y}_n$ is. It is an item of data that is taken at random and happens to be misclassified by the current value of $\boldsymbol{a}$ at iteration k.

Perceptron
Learning rule

- Gradient:

$$\frac{\partial E}{\partial \boldsymbol{a}} = -\sum \boldsymbol{y}_n$$

- Gradient algorithm: $\boldsymbol{a}^{(k+1)} = a^{(k)} + \sum \boldsymbol{y}_n$
- Stochastic gradient algorithm:

$$\boldsymbol{a}^{(k+1)} = a^{(k)} + \boldsymbol{y}_n$$

- Note what $\boldsymbol{y}_n$ is. It is an item of data that is taken at random and happens to be misclassified by the current value of $\boldsymbol{a}$ at iteration k.

Perceptron
Learning rule

- Gradient:

$$\frac{\partial E}{\partial \boldsymbol{a}} = -\sum \boldsymbol{y}_n$$

- Gradient algorithm: $\boldsymbol{a}^{(k+1)} = a^{(k)} + \sum \boldsymbol{y}_n$
- Stochastic gradient algorithm:

$$\boldsymbol{a}^{(k+1)} = a^{(k)} + \boldsymbol{y}_n$$

- Note what $\boldsymbol{y}_n$ is. It is an item of data that is taken at random and happens to be misclassified by the current value of $\boldsymbol{a}$ at iteration k.

- Gradient:

$$\frac{\partial E}{\partial \boldsymbol{a}} = - \sum \boldsymbol{y}_n$$

- Gradient algorithm: $\boldsymbol{a}^{(k+1)} = a^{(k)} + \sum \boldsymbol{y}_n$
- Stochastic gradient algorithm:

$$\boldsymbol{a}^{(k+1)} = a^{(k)} + \boldsymbol{y}_n$$

- Note what $\boldsymbol{y}_n$ is. It is an item of data that is taken at random and happens to be misclassified by the current value of $\boldsymbol{a}$ at iteration k.

Perceptron

Convergence of the learning rule

- Learning Rule: $\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} + \boldsymbol{y}(k)$
  where $\boldsymbol{y}(k)$ is a misclassified input.
- Training criterion
  - We start with requiring $\boldsymbol{a}^t\, \boldsymbol{y}(k) \lesssim 0$, depending on the example belonging to class 1 or class 2.
  - If we switch the signs of examples of class 2, we require $\boldsymbol{a}^t\, \boldsymbol{y}(k) > 0$ for all k.
- On misclassified data $\boldsymbol{a}^t\, \boldsymbol{y}(k) < 0$
- If $\widehat{\boldsymbol{a}}$ is a solution (separable data), for all k, $\widehat{\boldsymbol{a}}\, \boldsymbol{y}(k) > 0$
- We prove convergence by showing:
  $||\boldsymbol{a}^{(k+1)} - \widehat{\mathrm{a}}||^2 < ||\boldsymbol{a}^{(k)} - \widehat{\mathrm{a}}||^2$ for this update rule. *i.e.* the learning rule brings the guess closer to a valid solution.

- Learning Rule: $\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} + \boldsymbol{y}(k)$
  where $\boldsymbol{y}(k)$ is a misclassified input.
- Training criterion
  - We start with requiring $\boldsymbol{a}^t\, \boldsymbol{y}(k) \lessgtr 0$, depending on the example belonging to class 1 or class 2.
  - If we switch the signs of examples of class 2, we require $\boldsymbol{a}^t\, \boldsymbol{y}(k) > 0$ for all k.
- On misclassified data $\boldsymbol{a}^t\, \boldsymbol{y}(k) < 0$
- If $\widehat{\boldsymbol{a}}$ is a solution (separable data), for all k, $\widehat{\boldsymbol{a}}\, \boldsymbol{y}(k) > 0$
- We prove convergence by showing:
  $||\boldsymbol{a}^{(k+1)} - \widehat{\mathrm{a}}||^2 < ||\boldsymbol{a}^{(k)} - \widehat{\mathrm{a}}||^2$ for this update rule. *i.e.* the learning rule brings the guess closer to a valid solution.

Perceptron

Convergence of the learning rule

- Learning Rule: $\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} + \boldsymbol{y}(k)$
  where $\boldsymbol{y}(k)$ is a misclassified input.
- Training criterion
  - We start with requiring $\boldsymbol{a}^t\, \boldsymbol{y}(k) \lesseqgtr 0$, depending on the example belonging to class 1 or class 2.
  - If we switch the signs of examples of class 2, we require $\boldsymbol{a}^t\, \boldsymbol{y}(k) > 0$ for all k.
- On misclassified data $\boldsymbol{a}^t\, \boldsymbol{y}(k) < 0$
- If $\widehat{\boldsymbol{a}}$ is a solution (separable data), for all k, $\widehat{\boldsymbol{a}}\, \boldsymbol{y}(k) > 0$
- We prove convergence by showing:
  $||\boldsymbol{a}^{(k+1)} - \widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \widehat{\boldsymbol{a}}||^2$ for this update rule. *i.e.* the learning rule brings the guess closer to a valid solution.

Perceptron

Convergence of the learning rule

- Learning Rule: $\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} + \boldsymbol{y}(k)$
  where $\boldsymbol{y}(k)$ is a misclassified input.
- Training criterion
  - We start with requiring $\boldsymbol{a}^t \, \boldsymbol{y}(k) \lesseqgtr 0$, depending on the example belonging to class 1 or class 2.
  - If we switch the signs of examples of class 2, we require $\boldsymbol{a}^t \, \boldsymbol{y}(k) > 0$ for all k.
- On misclassified data $\boldsymbol{a}^t \, \boldsymbol{y}(k) < 0$
- If $\widehat{\boldsymbol{a}}$ is a solution (separable data), for all k, $\widehat{\boldsymbol{a}} \, \boldsymbol{y}(k) > 0$
- We prove convergence by showing:
  $||\boldsymbol{a}^{(k+1)} - \widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \widehat{\boldsymbol{a}}||^2$ for this update rule. *i.e.* the learning rule brings the guess closer to a valid solution.

- Learning Rule: $\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} + \boldsymbol{y}(k)$
  where $\boldsymbol{y}(k)$ is a misclassified input.
- Training criterion
  - We start with requiring $\boldsymbol{a}^t\,\boldsymbol{y}(k) \lessgtr 0$, depending on the example belonging to class 1 or class 2.
  - If we switch the signs of examples of class 2, we require $\boldsymbol{a}^t\,\boldsymbol{y}(k) > 0$ for all k.
- On misclassified data $\boldsymbol{a}^t\,\boldsymbol{y}(k) < 0$
- If $\widehat{\boldsymbol{a}}$ is a solution (separable data), for all k, $\widehat{\boldsymbol{a}}\,\boldsymbol{y}(k) > 0$
- We prove convergence by showing:
  $||\boldsymbol{a}^{(k+1)} - \widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \widehat{\boldsymbol{a}}||^2$ for this update rule. *i.e.* the learning rule brings the guess closer to a valid solution.

Perceptron

Convergence of the learning rule (cont'd)

- For perceptron criterion, the magnitude of $\boldsymbol{a}$ is not relevant (only the direction is). Hence for some scalar α, we wish to show

$$||\boldsymbol{a}^{(k+1)} - \alpha\,\widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \alpha\,\widehat{\boldsymbol{a}}||^2$$

- From the update formula

$$\boldsymbol{a}^{(k+1)} - \alpha\,\widehat{\boldsymbol{a}} = \boldsymbol{a}^{(k)} - \alpha\,\widehat{\boldsymbol{a}} + \boldsymbol{y}(k)$$

- Taking magnitudes

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 = ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 + 2(\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}})^t\boldsymbol{y}(k) + ||\boldsymbol{y}(k)||^2$$

- If we drop the negative term ${\boldsymbol{a}^{(k)}}^t\boldsymbol{y}(k)$ from RHS, the equality becomes an inequality

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 - 2\alpha\widehat{\boldsymbol{a}}^t\boldsymbol{y}(k) + ||\boldsymbol{y}(k)||^2$$

Perceptron

Convergence of the learning rule (cont'd)

- For perceptron criterion, the magnitude of $\boldsymbol{a}$ is not relevant (only the direction is). Hence for some scalar α, we wish to show

$$||\boldsymbol{a}^{(k+1)} - \alpha\,\widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \alpha\,\widehat{\boldsymbol{a}}||^2$$

- From the update formula

$$\boldsymbol{a}^{(k+1)} - \alpha\,\widehat{\boldsymbol{a}} = \boldsymbol{a}^{(k)} - \alpha\,\widehat{\boldsymbol{a}} + \boldsymbol{y}(k)$$

- Taking magnitudes

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 = ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 + 2(\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}})^t\boldsymbol{y}(k) + ||\boldsymbol{y}(k)||^2$$

- If we drop the negative term $\boldsymbol{a}^{(k)t}\boldsymbol{y}(k)$ from RHS, the equality becomes an inequality

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 - 2\alpha\widehat{\boldsymbol{a}}^t\boldsymbol{y}(k) + ||\boldsymbol{y}(k)||^2$$

Perceptron

Convergence of the learning rule (cont'd)

- For perceptron criterion, the magnitude of $\boldsymbol{a}$ is not relevant (only the direction is). Hence for some scalar α, we wish to show

$$||\boldsymbol{a}^{(k+1)} - \alpha\,\widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \alpha\,\widehat{\boldsymbol{a}}||^2$$

- From the update formula

$$\boldsymbol{a}^{(k+1)} - \alpha\,\widehat{\boldsymbol{a}} = \boldsymbol{a}^{(k)} - \alpha\,\widehat{\boldsymbol{a}} + \boldsymbol{y}(k)$$

- Taking magnitudes

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 = ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 + 2(\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}})^t\boldsymbol{y}(k) + ||\boldsymbol{y}(k)||^2$$

- If we drop the negative term ${\boldsymbol{a}^{(k)}}^t\boldsymbol{y}(k)$ from RHS, the equality becomes an inequality

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 - 2\alpha\widehat{\boldsymbol{a}}^t\boldsymbol{y}(k) + ||\boldsymbol{y}(k)||^2$$

- For perceptron criterion, the magnitude of $\boldsymbol{a}$ is not relevant (only the direction is). Hence for some scalar α, we wish to show

$$||\boldsymbol{a}^{(k+1)} - \alpha\,\widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \alpha\,\widehat{\boldsymbol{a}}||^2$$

- From the update formula

$$\boldsymbol{a}^{(k+1)} - \alpha\,\widehat{\boldsymbol{a}} = \boldsymbol{a}^{(k)} - \alpha\,\widehat{\boldsymbol{a}} + \boldsymbol{y}(k)$$

- Taking magnitudes

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 = ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 + 2(\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}})^t\boldsymbol{y}(k) + ||\boldsymbol{y}(k)||^2$$

- If we drop the negative term ${\boldsymbol{a}^{(k)}}^t\boldsymbol{y}(k)$ from RHS, the equality becomes an inequality

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 - 2\alpha\widehat{\boldsymbol{a}}^t\boldsymbol{y}(k) + ||\boldsymbol{y}(k)||^2$$

Perceptron

Convergence of the learning rule (cont'd)

- Of the three terms on the right hand side, we know $\widehat{\mathbf{a}}^t \mathbf{y}(k) > 0$, because $\widehat{\mathbf{a}}$ is assumed to be a solution.
- If we select

$$\beta^2 = \max_i ||\mathbf{y}_i||^2$$
$$\gamma = \min_i \widehat{\mathbf{a}}^t \mathbf{y}_i$$

  i.e. largest of the positive term and smallest of the negative term, then for $\alpha = \beta^2/\gamma$,

$$||\mathbf{a}^{(k+1)} - \alpha\widehat{\mathbf{a}}||^2 < ||\mathbf{a}^{(k)} - \alpha\widehat{\mathbf{a}}||^2 - \beta^2$$

- (Note the inequality remains true when the right hand side is replaced by a quantity larger than what it previously was.)
- Every correction takes the guess closer to a true solution.
- From an initialization $\mathbf{a}^{(1)}$, we will find a solution in *at most* $k_0 = \frac{||\mathbf{a}(1) - \alpha\widehat{\mathbf{a}}||^2}{\beta^2}$ updates.

Perceptron

Convergence of the learning rule (cont'd)

- Of the three terms on the right hand side, we know $\widehat{\boldsymbol{a}}^t \boldsymbol{y}(k) > 0$, because $\widehat{\boldsymbol{a}}$ is assumed to be a solution.
- If we select

$$\begin{aligned} \beta^2 &= \max_i ||\boldsymbol{y}_i||^2 \\ \gamma &= \min_i \widehat{\boldsymbol{a}}^t \boldsymbol{y}_i \end{aligned}$$

  i.e. largest of the positive term and smallest of the negative term, then for $\alpha = \beta^2/\gamma$,

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 - \beta^2$$

- (Note the inequality remains true when the right hand side is replaced by a quantity larger than what it previously was.)
- Every correction takes the guess closer to a true solution.
- From an initialization $\boldsymbol{a}^{(1)}$, we will find a solution in *at most* $k_0 = \frac{||\boldsymbol{a}(1) - \alpha\widehat{\boldsymbol{a}}||^2}{\beta^2}$ updates.

Perceptron

Convergence of the learning rule (cont'd)

- Of the three terms on the right hand side, we know $\widehat{\boldsymbol{a}}^t \boldsymbol{y}(k) > 0$, because $\widehat{\boldsymbol{a}}$ is assumed to be a solution.
- If we select

$$\begin{aligned} \beta^2 &= \max_i ||\boldsymbol{y}_i||^2 \\ \gamma &= \min_i \widehat{\boldsymbol{a}}^t \boldsymbol{y}_i \end{aligned}$$

  i.e. largest of the positive term and smallest of the negative term, then for $\alpha = \beta^2/\gamma$,

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 \; < \; ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 \; - \; \beta^2$$

- (Note the inequality remains true when the right hand side is replaced by a quantity larger than what it previously was.)
- Every correction takes the guess closer to a true solution.
- From an initialization $\boldsymbol{a}^{(1)}$, we will find a solution in *at most* $k_0 = \frac{||\boldsymbol{a}(1) - \alpha\widehat{\boldsymbol{a}}||^2}{\beta^2}$ updates.

Perceptron

Convergence of the learning rule (cont'd)

- Of the three terms on the right hand side, we know $\widehat{\boldsymbol{a}}^t \boldsymbol{y}(k) > 0$, because $\widehat{\boldsymbol{a}}$ is assumed to be a solution.
- If we select

$$\begin{aligned} \beta^2 &= \max_i ||\boldsymbol{y}_i||^2 \\ \gamma &= \min_i \widehat{\boldsymbol{a}}^t \boldsymbol{y}_i \end{aligned}$$

i.e. largest of the positive term and smallest of the negative term, then for $\alpha = \beta^2/\gamma$,

$$||\boldsymbol{a}^{(k+1)} - \alpha\widehat{\boldsymbol{a}}||^2 < ||\boldsymbol{a}^{(k)} - \alpha\widehat{\boldsymbol{a}}||^2 - \beta^2$$

- (Note the inequality remains true when the right hand side is replaced by a quantity larger than what it previously was.)
- Every correction takes the guess closer to a true solution.
- From an initialization $\boldsymbol{a}^{(1)}$, we will find a solution in *at most* $k_0 = \frac{||\boldsymbol{a}(1) - \alpha\widehat{\boldsymbol{a}}||^2}{\beta^2}$ updates.

Summary

- Linear regression
  - Solution as pseudo inverse
  - Solution by gradient descent
  - Regularization
- Perceptron
  - Setting up a suitable error function
  - Convergence of the algorithm