Outline

1. Administration
2. Review of last lecture
3. Linear regression
Outline

1. Administration

2. Review of last lecture
   - Perceptron
   - Linear regression

3. Linear regression
Perceptron algorithm

Iteratively solving one case at a time

- **REPEAT**
- Pick a data point $x_n$ (can be a fixed order of the training instances)
- Make a prediction $y = \text{sign}(\mathbf{w}^T \mathbf{x}_n)$
- If $y = y_n$, do nothing. Else,

$$
\mathbf{w} \leftarrow \mathbf{w} + y_n \mathbf{x}_n
$$

- **UNTIL** converged.
Linear regression

Setup

- Input: $x \in \mathbb{R}^D$ (covariates, predictors, features, etc)
- Output: $y \in \mathbb{R}$ (responses, targets, outcomes, outputs, etc)
- Training data: $\mathcal{D} = \{(x_n, y_n), n = 1, 2, \ldots, N\}$
- Model: $f : x \rightarrow y$, with $f(x) = w_0 + \sum_d w_d x_d = w_0 + \mathbf{w}^T \mathbf{x}$
- Objective: minimize prediction error as much as possible

$$RSS(\tilde{w}) = \sum_n [y_n - f(x_n)]^2 = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2$$
Probabilistic interpretation

- Noisy observation model

\[ Y = w_0 + w_1 X + \eta \]

where \( \eta \sim N(0, \sigma^2) \) is a Gaussian random variable

- Likelihood of one training sample \((x_n, y_n)\)

\[ p(y_n| x_n) = N(w_0 + w_1 x, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(y_n-(w_0+w_1 x_n))^2}{2\sigma^2}} \]
Maximum likelihood estimation

- Maximize over $w_0$ and $w_1$

$$\max \log P(\mathcal{D}) \Leftrightarrow \min \sum_n [y_n - (w_0 + w_1 x_n)]^2 \leftarrow \text{That is RSS}(\tilde{w})!$$
Maximum likelihood estimation

- Maximize over $w_0$ and $w_1$

$$\max \log P(D) \Leftrightarrow \min \sum \limits_n [y_n - (w_0 + w_1 x_n)]^2 \leftarrow \text{That is RSS}(\tilde{w})!$$

- Maximize over $s = \sigma^2$ (we could estimate $\sigma$ directly)

$$\frac{\partial \log P(D)}{\partial s} = -\frac{1}{2} \left\{ -\frac{1}{s^2} \sum \limits_n [y_n - (w_0 + w_1 x_n)]^2 + N \frac{1}{s} \right\} = 0$$
Maximum likelihood estimation

- Maximize over $w_0$ and $w_1$
  \[
  \max \log P(D) \iff \min \sum_n [y_n - (w_0 + w_1 x_n)]^2 \leftarrow \text{That is } \text{RSS}(\tilde{w})!
  \]

- Maximize over $s = \sigma^2$ (we could estimate $\sigma$ directly)
  \[
  \frac{\partial \log P(D)}{\partial s} = -\frac{1}{2} \left\{ -\frac{1}{s^2} \sum_n [y_n - (w_0 + w_1 x_n)]^2 + N \frac{1}{s} \right\} = 0
  \]
  \[
  \rightarrow \sigma^* = s^* = \frac{1}{N} \sum_n [y_n - (w_0 + w_1 x_n)]^2
  \]
Outline

1. Administration

2. Review of last lecture

3. Linear regression
   - Solution
   - Multivariate solution in matrix form
   - Computational and numerical optimization
   - Ridge regression
Solution when $x$ is one-dimensional

Least mean square (LMS) solution (minimizing residual sum of errors)

\[
\begin{pmatrix}
\sum_n n & \sum_n x_n \\
\sum_n x_n & \sum_n x_n^2
\end{pmatrix}
\begin{pmatrix}
w_0 \\
w_1
\end{pmatrix}
= 
\begin{pmatrix}
\sum_n y_n \\
\sum_n x_n y_n
\end{pmatrix}

\rightarrow
\begin{pmatrix}
w_0^{LMS} \\
w_1^{LMS}
\end{pmatrix}
= 
\left( \begin{pmatrix}
\sum_n n & \sum_n x_n \\
\sum_n x_n & \sum_n x_n^2
\end{pmatrix} \right)^{-1}
\begin{pmatrix}
\sum_n y_n \\
\sum_n x_n y_n
\end{pmatrix}

NB. We sometimes call it least square solutions too.
LMS when $\mathbf{x}$ is D-dimensional

**$RSS(\tilde{\mathbf{w}})$ in matrix form**

$$RSS(\tilde{\mathbf{w}}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_n]^2$$

where we have redefined some variables (by augmenting)

$$\tilde{\mathbf{x}} \leftarrow [1 \ x_1 \ x_2 \ldots \ x_D]^T, \quad \tilde{\mathbf{w}} \leftarrow [w_0 \ w_1 \ w_2 \ldots \ w_D]^T$$
LMS when $\mathbf{x}$ is D-dimensional

**RSS($\mathbf{\tilde{w}}$) in matrix form**

$$RSS(\mathbf{\tilde{w}}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \mathbf{\tilde{w}}^T \mathbf{\tilde{x}}_n]^2$$

where we have redefined some variables (by augmenting)

$$\mathbf{\tilde{x}} \leftarrow [1 \ x_1 \ x_2 \ \ldots \ x_D]^T, \quad \mathbf{\tilde{w}} \leftarrow [w_0 \ w_1 \ w_2 \ \ldots \ w_D]^T$$

which leads to

$$RSS(\mathbf{\tilde{w}}) = \sum_n (y_n - \mathbf{\tilde{w}}^T \mathbf{\tilde{x}}_n)(y_n - \mathbf{\tilde{x}}_n^T \mathbf{\tilde{w}})$$
LMS when $x$ is D-dimensional

**RSS($\tilde{\mathbf{w}}$) in matrix form**

$$RSS(\tilde{\mathbf{w}}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \tilde{\mathbf{w}}^T \tilde{x}_n]^2$$

where we have redefined some variables (by augmenting)

\[ \tilde{x} \leftarrow [1 \ x_1 \ x_2 \ \ldots \ x_D]^T, \quad \tilde{\mathbf{w}} \leftarrow [w_0 \ w_1 \ w_2 \ \ldots \ w_D]^T \]

which leads to

$$RSS(\tilde{\mathbf{w}}) = \sum_n (y_n - \tilde{\mathbf{w}}^T \tilde{x}_n)(y_n - \tilde{x}_n^T \tilde{\mathbf{w}})$$

$$= \sum_n \tilde{\mathbf{w}}^T \tilde{x}_n \tilde{x}_n^T \tilde{\mathbf{w}} - 2y_n \tilde{x}_n^T \tilde{\mathbf{w}} + \text{const.}$$
LMS when $x$ is D-dimensional

**RSS($\tilde{w}$) in matrix form**

$$RSS(\tilde{w}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \tilde{w}^T \tilde{x}_n]^2$$

where we have redefined some variables (by augmenting)

$$\tilde{x} \leftarrow [1 \ x_1 \ x_2 \ \ldots \ x_D]^T, \quad \tilde{w} \leftarrow [w_0 \ w_1 \ w_2 \ \ldots \ w_D]^T$$

which leads to

$$RSS(\tilde{w}) = \sum_n (y_n - \tilde{w}^T \tilde{x}_n)(y_n - \tilde{x}_n^T \tilde{w})$$

$$= \sum_n \tilde{w}^T \tilde{x}_n \tilde{x}_n^T \tilde{w} - 2y_n \tilde{x}_n^T \tilde{w} + \text{const.}$$

$$= \left\{ \tilde{w}^T \left( \sum_n \tilde{x}_n \tilde{x}_n^T \right) \tilde{w} - 2 \left( \sum_n y_n \tilde{x}_n^T \right) \tilde{w} \right\} + \text{const.}$$
$RSS(\tilde{w})$ in new notations

Design matrix and target vector

$$X = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} \in \mathbb{R}^{N \times D}, \quad \tilde{X} = (1 \quad X) \in \mathbb{R}^{N \times (D+1)}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$
$RSS(\tilde{w})$ in new notations

Design matrix and target vector

$$X = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} \in \mathbb{R}^{N \times D}, \quad \tilde{X} = \begin{pmatrix} 1 \\ X \end{pmatrix} \in \mathbb{R}^{N \times (D+1)}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

Compact expression

$$RSS(\tilde{w}) = \left\{ \tilde{w}^T \tilde{X}^T \tilde{X} \tilde{w} - 2 (\tilde{X}^T y)^T \tilde{w} \right\} + \text{const}$$
Solution in matrix form

**Normal equation**

Take derivative with respect to \( \tilde{w} \)

\[
\frac{\partial \text{RSS}(\tilde{w})}{\partial \tilde{w}} \propto \tilde{X}^T \tilde{X} \tilde{w} - \tilde{X}^T y = 0
\]

This leads to the least-mean-square (LMS) solution

\[
\tilde{w}^{LMS} = \left( \tilde{X}^T \tilde{X} \right)^{-1} \tilde{X}^T y
\]
Solution in matrix form

**Normal equation**
Take derivative with respect to $\tilde{w}$

$$\frac{\partial RSS(\tilde{w})}{\partial \tilde{w}} \propto \tilde{X}^T \tilde{X} \tilde{w} - \tilde{X}^T y = 0$$

This leads to the least-mean-square (LMS) solution

$$\tilde{w}^{LMS} = \left( \tilde{X}^T \tilde{X} \right)^{-1} \tilde{X}^T y$$

**Verify the solution when $D = 1$**

$$\tilde{X}^T \tilde{X} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_N \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdots & \cdots \\ 1 & x_N \end{pmatrix} = \begin{pmatrix} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{pmatrix}$$
Linear regression is the linear combination of features.
\[ f : \mathbf{x} \to y, \text{ with } f(x) = w_0 + \sum_d w_d x_d = w_0 + \mathbf{w}^T \mathbf{x} \]

If we minimize residual sum squares as our learning objective, we get a closed-form solution of parameters.

Probabilistic interpretation: maximum likelihood if assuming residual is Gaussian distributed.

Other interpretations exist: if interested, please consult the slides from last year’s lectures.
Computational complexity

Bottleneck of computing the solution

\[ w = \left( \tilde{X}^T \tilde{X} \right)^{-1} \tilde{X} y \]

is to invert the matrix \( \tilde{X}^T \tilde{X} \in \mathbb{R}^{(D+1) \times (D+1)} \)

How many operations do we need?

- On the order of \( O((D + 1)^3) \) (using Gauss-Jordan elimination) or \( O((D + 1)^{2.373}) \) (recent advances in computing)
- Impractical for very large \( D \)
Alternative method: an example of using numerical optimization

*(Batch) Gradient descent*

- Initialize $\tilde{w}$ to $\tilde{w}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
Alternative method: an example of using numerical optimization

(Batch) Gradient descent

- Initialize $\tilde{w}$ to $\tilde{w}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop until convergence
  - Compute the gradient (ignoring the constant factor)
    \[ \nabla RSS(\tilde{w}) = \tilde{X}^T \tilde{X} \tilde{w}^{(t)} - \tilde{X}^T y \]
Alternative method: an example of using numerical optimization

(Batch) Gradient descent

- Initialize $\tilde{w}$ to $\tilde{w}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop until convergence
  1. Compute the gradient (ignoring the constant factor)
     $\nabla RSS(\tilde{w}) = \tilde{X}^T \tilde{X} \tilde{w}^{(t)} - \tilde{X}^T y$
  2. Update the parameters
     $\tilde{w}^{(t+1)} = \tilde{w}^{(t)} - \eta \nabla RSS(\tilde{w})$
Alternative method: an example of using numerical optimization

**(Batch) Gradient descent**

- Initialize \( \tilde{w} \) to \( \tilde{w}^{(0)} \) (anything reasonable is fine); set \( t = 0 \); choose \( \eta > 0 \)
- Loop *until convergence*
  - Compute the gradient (ignoring the constant factor)
    \[
    \nabla RSS(\tilde{w}) = \tilde{X}^T \tilde{X} \tilde{w}^{(t)} - \tilde{X}^T y
    \]
  - Update the parameters
    \[
    \tilde{w}^{(t+1)} = \tilde{w}^{(t)} - \eta \nabla RSS(\tilde{w})
    \]
  - \( t \leftarrow t + 1 \)
Alternative method: an example of using numerical optimization

(Batch) Gradient descent

- Initialize $\tilde{w}$ to $\tilde{w}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop until convergence
  1. Compute the gradient (ignoring the constant factor)
     \[ \nabla RSS(\tilde{w}) = \tilde{X}^T \tilde{X} \tilde{w}^{(t)} - \tilde{X}^T y \]
  2. Update the parameters
     \[ \tilde{w}^{(t+1)} = \tilde{w}^{(t)} - \eta \nabla RSS(\tilde{w}) \]
  3. $t \leftarrow t + 1$

What is the complexity here?
Why would this work?

If gradient descent converges, it will converge to the same solution using matrix inversion.

This is because $RSS(\tilde{w})$ is a convex function in its parameters $w$

$$RSS(\tilde{w}) = \tilde{w}^T \tilde{X}^T \tilde{X} \tilde{w} - 2 (\tilde{X}^T y)^T \tilde{w}$$
Why would this work?

If gradient descent converges, it will converge to the same solution using matrix inversion.

This is because $RSS(\tilde{w})$ is a convex function in its parameters $w$

$$RSS(\tilde{w}) = \tilde{w}^T \tilde{X}^T \tilde{X} \tilde{w} - 2 (\tilde{X}^T y)^T \tilde{w}$$

$$\Rightarrow \frac{\partial^2 RSS(\tilde{w})}{\partial \tilde{w} \tilde{w}^T} = 2 \tilde{X}^T \tilde{X}$$
Why would this work?

If gradient descent converges, it will converge to the same solution using matrix inversion.

This is because $RSS(\tilde{w})$ is a convex function in its parameters $w$

$$RSS(\tilde{w}) = \tilde{w}^T \tilde{X}^T \tilde{X} \tilde{w} - 2 \left( \tilde{X}^T y \right)^T \tilde{w}$$

$$\Rightarrow \frac{\partial^2 RSS(\tilde{w})}{\partial \tilde{w} \tilde{w}^T} = 2 \tilde{X}^T \tilde{X}$$

as $\tilde{X}^T \tilde{X}$ is positive semidefinite, because for any $v$

$$v^T \tilde{X}^T \tilde{X} v = \| \tilde{X}^T v \|_2^2 \geq 0$$
Stochastic gradient descent

**Widrow-Hoff rule**: update parameters using one example at a time

- Initialize $\tilde{w}$ to $\tilde{w}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
Stochastic gradient descent

**Widrow-Hoff rule**: update parameters using one example at a time

- Initialize $\tilde{w}$ to $\tilde{w}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. random choose a training a sample $x_t$
  2. Compute its contribution to the gradient (ignoring the constant factor)

$$g_t = (\tilde{x}_t^T \tilde{w}^{(t)} - y_t)\tilde{x}_t$$
Stochastic gradient descent

**Widrow-Hoff rule**: update parameters using one example at a time

- Initialize $\tilde{w}$ to $\tilde{w}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop until convergence
  1. Randomly choose a training sample $x_t$
  2. Compute its contribution to the gradient (ignoring the constant factor)
     \[
     g_t = (\tilde{x}_t^T \tilde{w}^{(t)} - y_t) \tilde{x}_t
     \]
  3. Update the parameters
     \[
     \tilde{w}^{(t+1)} = \tilde{w}^{(t)} - \eta g_t
     \]
Stochastic gradient descent

**Widrow-Hoff rule**: update parameters using one example at a time

- Initialize $\tilde{w}$ to $\tilde{w}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop until convergence
  1. Randomly choose a training sample $x_t$
  2. Compute its contribution to the gradient (ignoring the constant factor)

\[
g_t = (\tilde{x}_t^T \tilde{w}^{(t)} - y_t) \tilde{x}_t
\]

- Update the parameters
  \[
  \tilde{w}^{(t+1)} = \tilde{w}^{(t)} - \eta g_t
  \]
- $t \leftarrow t + 1$
Stochastic gradient descent

**Widrow-Hoff rule**: update parameters using one example at a time

- Initialize $\tilde{\mathbf{w}}$ to $\tilde{\mathbf{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. Randomly choose a training sample $\mathbf{x}_t$
  2. Compute its contribution to the gradient (ignoring the constant factor)

\[
g_t = (\tilde{x}_t^T \tilde{\mathbf{w}}^{(t)} - y_t)\tilde{x}_t
\]

3. Update the parameters

\[
\tilde{\mathbf{w}}^{(t+1)} = \tilde{\mathbf{w}}^{(t)} - \eta g_t
\]

4. $t \leftarrow t + 1$

*What is the complexity here?*
Mini-summary

- Batch gradient descent computes the exact gradient.
- Stochastic gradient descent computes the gradient pretending only one instance.
  Its expectation equals to the true gradient.
- Other forms can be used.
  Mini-batch: trade-off between accuracy of estimating gradient and computational cost
- Similar ideas extend to other optimization problems in machine learning.
  For large-scale problems, stochastic gradient descent often works well.
What if $\tilde{X}^T \tilde{X}$ is not invertible

Can you think of any reasons why that could happen?

Answer 1: $N < D$. Intuitively, not enough data to estimate all the parameters.

Answer 2: $X$ columns are not linearly independent. Intuitively, there are two features that are perfectly correlated. In this case, solution is not unique.
What if $\tilde{X}^T \tilde{X}$ is not invertible

Can you think of any reasons why that could happen?

**Answer 1:** $N < D$. Intuitively, not enough data to estimate all the parameters.
What if $\tilde{X}^T\tilde{X}$ is not invertible

Can you think of any reasons why that could happen?

**Answer 1:** $N < D$. Intuitively, not enough data to estimate all the parameters.

**Answer 2:** $X$ columns are not linearly independent. Intuitively, there are two features that are perfectly correlated. In this case, solution is not unique.
**Intuition:** what does a non-invertible $\tilde{X}^T \tilde{X}$ mean?

$$\tilde{X}^T \tilde{X} = U^T \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \lambda_r & 0 \\ 0 & \cdots & \cdots & 0 & 0 \end{bmatrix} U$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_r > 0$ and $r < D$. 
Ridge regression

**Intuition:** what does a non-invertible $\tilde{X}^T \tilde{X}$ mean?

$$
\tilde{X}^T \tilde{X} = U^T \begin{bmatrix}
\lambda_1 & 0 & 0 & \cdots & 0 \\
0 & \lambda_2 & 0 & \cdots & 0 \\
0 & \cdots & \cdots & \cdots & 0 \\
0 & \cdots & \cdots & \lambda_r & 0 \\
0 & \cdots & \cdots & 0 & 0
\end{bmatrix} U
$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_r > 0$ and $r < D$.

**Fix the problem** by adding something positive

$$
\tilde{X}^T \tilde{X} + \lambda I = U^T \text{diag}(\lambda_1 + \lambda, \lambda_2 + \lambda, \cdots, \lambda) U
$$

where $\lambda > 0$ and $I$ is the identity matrix.
Regularized least square (ridge regression)

Solution

$$\tilde{w} = \left( \tilde{X}^T \tilde{X} + \lambda I \right)^{-1} \tilde{X}^T y$$
Regularized least square (ridge regression)

**Solution**

\[ \tilde{w} = \left( \tilde{X}^T \tilde{X} + \lambda I \right)^{-1} \tilde{X}^T y \]

This is equivalent to adding an extra term to \( RSS(\tilde{w}) \)

\[ \frac{1}{2} \left\{ \tilde{w}^T \tilde{X}^T \tilde{X} \tilde{w} - 2 \left( \tilde{X}^T y \right)^T \tilde{w} \right\} + \frac{1}{2} \lambda \| \tilde{w} \|_2^2 \]

regularization
Regularized least square (ridge regression)

Solution

\[ \tilde{w} = \left( \tilde{X}^T \tilde{X} + \lambda I \right)^{-1} \tilde{X}^T y \]

This is equivalent to adding an extra term to \( RSS(\tilde{w}) \)

\[ RSS(\tilde{w}) \]

\[ \frac{1}{2} \left\{ \tilde{w}^T \tilde{X}^T \tilde{X} \tilde{w} - 2 \left( \tilde{X}^T y \right)^T \tilde{w} \right\} + \frac{1}{2} \lambda \| \tilde{w} \|_2^2 \]  

regularization

Benefits

- Numerically more stable, invertible matrix
- Prevent overfitting — more on this later
How to choose $\lambda$?

Again, $\lambda$ is referred as *hyperparameter*, to be distinguished from $w$.

- Use validation or cross-validation
- Other approaches such as Bayesian linear regression — we will describe them briefly later