Linear regression

• Shorter (vector) definition of the model
  – Include bias constant in the input vector

\[ x = (1, x_1, x_2, \ldots, x_d) \]

\[ f(x) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d = w^T x \]

\[ w = [w_0, w_1, \ldots, w_k]^T \] - parameters (weights)

\[ \sum \]

Input vector \( x \) 

\[ f(x, w) \]
Linear regression. Example

- 1 dimensional input \( x = (x_1) \)

\[
\begin{array}{c|c|c|c|c|c|c}
-1.5 & -1 & -0.5 & 0 & 0.5 & 1 & 1.5 & 2 \\
-15 & -10 & -5 & 0 & 5 & 10 & 15 & 20 \\
\end{array}
\]

Linear regression. Example.

- 2 dimensional input \( x = (x_1, x_2) \)

\[
\begin{array}{c|c|c|c|c|c|c}
-3 & -2 & -1 & 0 & 1 & 2 & 3 \\
-4 & -2 & 0 & 2 & 4 & 6 & 8 \\
-15 & -10 & -5 & 0 & 5 & 10 & 15 \\
\end{array}
\]
Linear regression: error

- Data: \( D_i = \langle x_i, y_i \rangle \)
- Function: \( x_i \to f(x_i) \)
- Error: a measure of misfit of the model and the data
  \[
  J_n = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2
  \]

Solving linear regression

- The optimal set of weights satisfies:
  \[
  \nabla_w (J_n (w)) = -2 \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)x_i = 0
  \]
  Leads to a system of linear equations (SLE) with \( d+1 \) unknowns of the form
  \[
  Aw = b
  \]
  \[
  w_0 \sum_{i=1}^{n} x_{i,0}x_{i,j} + w_1 \sum_{i=1}^{n} x_{i,1}x_{i,j} + \ldots + w_j \sum_{i=1}^{n} x_{i,j}x_{i,j} + \ldots + w_d \sum_{i=1}^{n} x_{i,d}x_{i,j} = \sum_{i=1}^{n} y_i x_{i,j}
  \]
  Solution to SLE:
  \[
  w = A^{-1} b
  \]
  Assuming \( X \) is an \( nxn \) data matrix with rows corresponding to examples and columns to inputs, and \( y \) is \( nx1 \) vector of outputs, then
  \[
  w = (X^T X)^{-1} X^T y
  \]
Gradient descent solution

**Objective:** optimize the weights in the linear regression model

\[ J_n = \text{Error}(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 \]

An alternative to SLE solution:

- **Gradient descent**
  
  Idea:
  
  - Adjust weights in the direction that improves the Error
  - The gradient tells us what is the right direction

\[ w \leftarrow w - \alpha \nabla_w \text{Error}_i(w) \]

\[ \alpha > 0 \quad \text{a learning rate} \quad \text{(scales the gradient changes)} \]

Batch vs online gradient algorithm

- The error function defined on the complete dataset \( D \)

\[ J_n = \text{Error}(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 \]

- We say we are learning the model in **the batch mode**:
  
  - All examples are available at the time of learning
  - Weights are optimized with respect to all training examples

- An alternative is to learn the model in **the online mode**

\[ J_{\text{online}} = \text{Error}_i(w, x_i) = \frac{1}{2} (y_i - f(x_i, w))^2 \]

- Examples are arriving sequentially
- Model weights are updated after every example
- If needed examples seen can be forgotten
Online gradient descent algorithm

**Online-linear-regression** (*stopping_criterion*)
- Initialize weights \( w = (w_0, w_1, w_2 \ldots w_d) \)
- initialize \( i=1; \)
  - while \( stopping_criterion = FALSE \)
    - select the next data point \( D_i = (x_i, y_i) \)
    - set learning rate \( \alpha(i) \)
    - update weight vector \( w \leftarrow w + \alpha(i)(y_i - f(x, w))x_i \)
  - end
- return weights

**Advantages:** very easy to implement, works on continuous data streams

---

Extensions of simple linear model

Replace inputs to linear units with \( m \) feature (basis) functions to model nonlinearities

\[
f(x) = w_0 + \sum_{j=1}^{m} w_j \phi_j(x)
\]

\( \phi_j(x) \) - an arbitrary function of \( x \)

Original input \[\rightarrow\] New input \[\rightarrow\] Linear model
Non-linear (quadratic) model

Linear regression model

- Linear model: \( y = f(x, w) = w^T x \)

- Notice: the above model does not explain variation in observed ys for the data
**Statistical model of regression**

A statistical model of linear regression:

\[ y = w^T x + \varepsilon \quad \varepsilon \sim N(0, \sigma^2) \]

\( \varepsilon \) is a random noise, represents deviations not captured with \( w^T x \)

\[ y \sim N(w^T x, \sigma^2) \]
Statistical model of regression

A statistical model of linear regression:

\[ y = w^T x + \varepsilon \quad \varepsilon \sim N(0, \sigma^2) \]

\[ y \sim N(w^T x, \sigma^2) \]

- The conditional distribution of \( y \) given \( x \)

\[ p(y \mid x, w, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left[ -\frac{1}{2\sigma^2} (y - w^T x)^2 \right] \]

\[ E(y \mid x) = w^T x \]

ML estimation of the parameters

- **likelihood of predictions** = the probability of observing outputs \( y \) in \( D \) given \( w, \sigma \)

\[ L(D, w, \sigma) = \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma) \]

- **Maximum likelihood estimation of parameters** \( w \)
  - parameters maximizing the likelihood of predictions

\[ w^* = \arg \max_w \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma) \]

- **Log-likelihood** trick for the ML optimization
  - Maximizing the log-likelihood is equivalent to maximizing the likelihood

\[ l(D, w, \sigma) = \log(L(D, w, \sigma)) = \log \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma) \]
ML estimation of the parameters

- **Using conditional density**
  
  \[ p(y \mid x, w, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left[-\frac{1}{2\sigma^2} (y - f(x, w))^2 \right] \]

- **We can rewrite the log-likelihood as**
  
  \[
  l(D, w, \sigma) = \log( L(D, w, \sigma)) = \log \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma) \\
  = \sum_{i=1}^{n} \log p(y_i \mid x_i, w, \sigma) = \sum_{i=1}^{n} \left\{ - \frac{1}{2\sigma^2} (y_i - w^T x_i)^2 - c(\sigma) \right\} \\
  = - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + C(\sigma)
  \]

  Did we see a similar expression before?

---

ML estimation of the parameters

- **Using conditional density**
  
  \[ p(y \mid x, w, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left[-\frac{1}{2\sigma^2} (y - f(x, w))^2 \right] \]

- **We can rewrite the log-likelihood as**
  
  \[
  l(D, w, \sigma) = \log( L(D, w, \sigma)) = \log \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma) \\
  = \sum_{i=1}^{n} \log p(y_i \mid x_i, w, \sigma) = \sum_{i=1}^{n} \left\{ - \frac{1}{2\sigma^2} (y_i - w^T x_i)^2 - c(\sigma) \right\} \\
  = - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + C(\sigma)
  \]

- **Maximizing the predictive log likelihood** with regard to \( w \), is **equivalent to minimizing the mean squared error function**
**ML estimation of parameters**

- Criteria based on the mean squares error function and the log likelihood of the output are related
  \[ J_{\text{online}}(y_i, x_i) = \frac{1}{2\sigma^2} \log p(y_i \mid x_i, w, \sigma) + c(\sigma) \]
- We know how to optimize parameters \( w \)
  - the same approach as used for the least squares fit
- But what is the ML estimate of the variance of the noise?
- Maximize \( I(D, w, \sigma) \) with respect to variance
  \[ \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w^*))^2 \]
  \[ = \text{mean square prediction error for the best predictor} \]

---

**Regularized linear regression**

- If the number of parameters is large relative to the number of data points used to train the model, we face the threat of **overfitting** (generalization error of the model goes up)
- The prediction accuracy can be often improved by setting some coefficients (weights) of the model to zero
  - Increases the bias, reduces the variance of estimates
- **Solutions**:
  - Subset selection
  - Ridge regression
  - Lasso regression
  - Principal component regression
Regularization: motivation

- If the model is too complex and can cause overfitting, its prediction accuracy can be improved by removing some inputs from the model = setting their coefficients to zero.

\[ f(x) = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots w_d x_d = \mathbf{w}^T \mathbf{x} \]

\( w_0, w_1, \ldots, w_k \) - parameters (weights)

Input vector \( \mathbf{x} \)

Ridge regression

**Question:** how to force the weights to 0?

- Error function for the standard least squares estimates:

\[ J_n(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \]

- We seek: \( \mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \)

- Ridge regression:

\[ J_n(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \| \mathbf{w} \|_{L_2}^2 \]

**Fit to data**  \( \text{Model complexity penalty} \)

- Where \( \| \mathbf{w} \|_{L_2}^2 = \sum_{i=0}^{d} w_i^2 \) and \( \lambda \geq 0 \)

- What does the new error function do?
Ridge regression

Ridge regression:

\[ J_n(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda \|w\|_{L2}^2 \]

Term \( \|w\|_{L2}^2 = \sum_{i=0}^{d} w_i^2 \)

- penalizes non-zero weights with the cost that is proportional to \( \lambda \) (a shrinkage coefficient)
- If an input attribute \( x_j \) has a small effect on improving the error function it is “shut down” (driven to 0) by the penalty term
- Inclusion of a shrinkage penalty is often referred to as regularization.

Regularized linear regression

How to solve the least squares problem if the error function is enriched by the regularization term \( \lambda \|w\|_{L2}^2 \)?

**Answer:** The solution to the optimal set of weights \( w \) is obtained again by solving a set of linear equation.

**Standard linear regression:**

\[ \nabla_w (J_n (w)) = -\frac{2}{n} \sum_{i=1}^{n} (y_i - w^T x_i) x_i = \mathbf{0} \]

**Solution:** \( w^* = (X^TX)^{-1} X^T y \)

where \( X \) is an \( nxd \) matrix with rows corresponding to examples and columns to inputs

**Regularized linear regression:**

\( w^* = (\lambda I + X^TX)^{-1} X^T y \)
Lasso regression

- **Standard regression:**
  \[ J_n(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 \]

- **Lasso regression/regularization:**
  \[
  J_n(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda \|w\|_{L1}
  \]

  Fit to data | Model complexity penalty
  \[
  \|w\|_{L1} = \sum_{i=0}^{d} |w_i|
  \]
  and \( \lambda \geq 0 \)

- L1 is more aggressive pushing the weights to 0 compared to L2

Lasso vs Ridge penalty

- Lasso (L1) penalty \( \|w\|_{L1} = \sum_{i=0}^{d} |w_i| \)

- Ridge (L2) penalty \( \|w\|^2_{L2} = \sum_{i=0}^{d} w_i^2 \)

- L1 is more aggressive pushing the weights to 0 compared to L2