Linear regression

- **Function** \( f : X \rightarrow Y \) is a linear combination of input components
  \[
  f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_dx_d = w_0 + \sum_{j=1}^{d} w_jx_j
  \]
  \( w_0, w_1, \ldots, w_k \) - parameters (weights)
Linear regression. Error.

• Data:  \( D = \{x_i, y_i\} \)

• Function:  \( x_i \rightarrow f(x_i) \)

• We would like to have  \( y_i \approx f(x_i) \) for all \( i = 1, \ldots, n \)

• Error function
  – measures how much our predictions deviate from the desired answers
  
  \[
  \text{Mean-squared error} \quad J_n = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2
  \]

• Learning:
  We want to find the weights minimizing the error !

Linear regression. Example

• 1 dimensional input  \( x = (x_1) \)
Linear regression. Example.

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

Solving linear regression

- The optimal set of weights satisfies:
  \[
  \nabla_w J_n(w) = -\frac{2}{n} \sum_{i=1}^{n} (y_i - w^T \mathbf{x}_i) \mathbf{x}_i = \mathbf{0}
  \]

  Leads to a system of linear equations (SLE) with \( d+1 \)

unknowns of the form

\[
\mathbf{A} \mathbf{w} = \mathbf{b}
\]

Solution to SLE:

\[
\mathbf{w} = \mathbf{A}^{-1} \mathbf{b}
\]

- matrix inversion
Gradient descent solution

Goal: the weight optimization in the linear regression model

\[ J_n = \text{Error} (w) = \frac{1}{n} \sum_{j=1}^{n} (y_j - f(x_j, w))^2 \]

Iterative solution:

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

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

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

---

Gradient descent method

- Descend using the gradient information

\[ \text{Error} (w) \]

- Change the value of \( w \) according to the gradient

\[
    w \leftarrow w - \alpha \nabla_w \text{Error}_j(w)
\]
Gradient descent method

- Iteratively approaches the optimum of the Error function

\[ Error(w) \]

\[ w^{(0)} \rightarrow w^{(1)} \rightarrow w^{(2)} \rightarrow w^{(3)} \]

Online gradient method

Linear model \[ f(x) = w^T x \]

On-line error \[ J_{\text{online}} = Error_i(w) = \frac{1}{2} (y_i - f(x_i, w))^2 \]

On-line algorithm: generates a sequence of online updates

(i)-th update step with: \[ D_i = \langle x_i, y_i \rangle \]

j-th weight:

\[ w_j^{(i)} \leftarrow w_j^{(i-1)} - \alpha(i) \frac{\partial Error_i(w)}{\partial w_j} \bigg|_{w^{(i-1)}} \]

\[ w_j^{(i)} \leftarrow w_j^{(i-1)} + \alpha(i)(y_i - f(x_i, w^{(i-1)}))x_{i,j} \]

Fixed learning rate: \( \alpha(i) = C \)  
Annealed learning rate: \( \alpha(i) \approx \frac{1}{i} \)

- Use a small constant  
- Gradually rescales changes
Extensions of simple linear model

Replace inputs to linear units with **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 \( x \)

The same techniques as before to learn the weights
Additive linear models

- Models linear in the parameters we want to fit
  \[ f(x) = w_0 + \sum_{k=1}^{m} w_k \phi_k(x) \]

  \( w_0, w_1 \ldots w_m \) - parameters
  \( \phi_1(x), \phi_2(x) \ldots \phi_m(x) \) - feature or basis functions

- Basis functions examples:
  - a higher order polynomial, one-dimensional input \( x = (x_1) \)
    \( \phi_1(x) = x \quad \phi_2(x) = x^2 \quad \phi_3(x) = x^3 \)
  - Multidimensional quadratic \( x = (x_1, x_2) \)
    \( \phi_1(x) = x_1 \quad \phi_2(x) = x_1^2 \quad \phi_3(x) = x_2 \quad \phi_4(x) = x_2^2 \quad \phi_5(x) = x_1x_2 \)
  - Other types of basis functions
    \( \phi_1(x) = \sin x \quad \phi_2(x) = \cos x \)

Fitting additive linear models

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

  Assume:
  \( \varphi(x_i) = (1, \phi_1(x_i), \phi_2(x_i), \ldots, \phi_m(x_i)) \)

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

- Leads to a system of \( m \) linear equations

  \[ w_0 \sum_{i=1}^{n} \phi_j(x_i) + \ldots + w_j \sum_{i=1}^{n} \phi_j(x_i) \phi_j(x_i) + \ldots + w_m \sum_{i=1}^{n} \phi_m(x_i) \phi_j(x_i) = \sum_{i=1}^{n} y_i \phi_j(x_i) \]

- Can be solved exactly like the linear case
Example. Regression with polynomials.

Regression with polynomials of degree $m$

- **Data points:** pairs of $<x, y>$
- **Feature functions:** $m$ feature functions
  $$
  \phi_i(x) = x^i \quad i = 1, 2, \ldots, m
  $$
- **Function to learn:**
  $$
  f(x, w) = w_0 + \sum_{i=1}^{m} w_i \phi_i(x) = w_0 + \sum_{i=1}^{m} w_i x^i
  $$

Learning with feature functions.

**Function to learn:**
$$
  f(x, w) = w_0 + \sum_{i=1}^{k} w_i \phi_i(x)
  $$

**On line gradient update** for the $<x, y>$ pair
$$
  w_0 = w_0 + \alpha(y - f(x, w))
  $$
$$
  \vdots
  $$
$$
  w_j = w_j + \alpha(y - f(x, w)) \phi_j(x)
  $$

Gradient updates are of the same form as in the linear and logistic regression models.
Example. Regression with polynomials.

**Example:** Regression with polynomials of degree \( m \)

\[
f(x, w) = w_0 + \sum_{i=1}^{m} w_i \phi_i(x) = w_0 + \sum_{i=1}^{m} w_i x^i
\]

- **On line update** for \(<x, y>\) pair

  \[
  w_0 = w_0^0 + \alpha (y - f(x, w))
  \]

  \[
  \vdots
  \]

  \[
  w_j = w_j^0 + \alpha (y - f(x, w)) x^i
  \]

---

**Multidimensional additive model example**
Statistical model of regression

- A generative model: \( y = f(x, w) + \varepsilon \)
  
  \( f(x, w) \) is a deterministic function
  
  \( \varepsilon \) is a random noise, it represents things we cannot capture with \( f(x, w) \), e.g. \( \varepsilon \sim \mathcal{N}(0, \sigma^2) \)
**Statistical model of regression**

- **Assume a generative model:**
  \[ y = f(x, w) + \varepsilon \]
  where \( f(x, w) = w^T x \) is a linear model,
  and \( \varepsilon \sim N(0, \sigma^2) \)

- Then: \( f(x, w) = E(y | x) \)
  - models the mean of outputs \( y \) for \( x \)
  - and the **noise** \( \varepsilon \) models deviations from the mean

- **The model defines the conditional density** of \( y \) given \( x, w, \sigma \)

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

---

**ML estimation of the parameters**

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

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

- **Maximum likelihood estimation of parameters**
  - parameters maximizing the likelihood of predictions
  \[
w^* = \arg\max_w \prod_{i=1}^{n} p(y_i | x_i, w, \sigma)
\]

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

\[
l(D, w, \sigma) = \log L(D, w, \sigma) = \log \prod_{i=1}^{n} p(y_i | x_i, w, \sigma)
\]
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
  \[ l(D, w, \sigma) = \log(L(D, w, \sigma)) = \log \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma) \]
  \[ = \sum_{i=1}^{n} \log p(y_i \mid x_i, w, \sigma) = \sum_{i=1}^{n} \left\{ -\frac{1}{2\sigma^2} (y_i - f(x_i, w))^2 - c(\sigma) \right\} \]
  \[ = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 + C(\sigma) \]

- Maximizing with regard to \( w \), is equivalent to minimizing squared error function

ML estimation of parameters

- Criteria based on 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 \( l(D, w, \sigma) \) with respect to variance
  \[ \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w^*))^2 \]
  \[ = \text{mean squared prediction error for the best predictor} \]
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 overfit (generalization error of the model goes up)
- The prediction accuracy can be often improved by setting some coefficients to zero
  - Increases the bias, reduces the variance of estimates

**Solutions:**
- Subset selection
- Ridge regression
- Principal component regression

- Next: ridge regression

---

Ridge regression

- Error function for the standard least squares estimates:
  \[ J(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 \]
- We seek: \[ w^* = \arg \min_w \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 \]
- Ridge regression:
  \[ J_{\lambda}(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda \|w\|^2 \]
- Where \[ \|w\|^2 = \sum_{i=0}^{d} w_i^2 \] and \( \lambda \geq 0 \)
- What does the new error function do?
Ridge regression

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

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

  - \( \|w\|^2 = \sum_{i=0}^{d} w_i^2 \) penalizes non-zero weights with the cost 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” 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 \( \lambda \|w\|^2 \)?

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

**Standard linear regression:**

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

**Solution:**

\[ w^* = (X^T X)^{-1} X^T y \]

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

**Regularized linear regression:**

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