Introduction to Machine Learning

Linear Regression

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)
**Basic Supervised Learning Pipeline**

- **Training Data**
  - "spam"
  - "ham"
  - "spam"

- **Learning method**

- **Model fitting**

- **Test Data**

- **Prediction**

- **Prediction/Generalization**

\[
\mathbf{x} \rightarrow f : \mathbf{x} \rightarrow \mathbf{y}
\]
Regression

- Instance of supervised learning
- **Goal**: Predict *real valued* labels (possibly vectors)
- Examples:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight route</td>
<td>Delay (minutes)</td>
</tr>
<tr>
<td>Real estate objects</td>
<td>Price</td>
</tr>
<tr>
<td>Customer &amp; ad features</td>
<td>Click-through probability</td>
</tr>
</tbody>
</table>
Running example: Diabetes

[Efron et al ‘04]

- **Features X:**
  - Age
  - Sex
  - Body mass index
  - Average blood pressure
  - Six blood serum measurements (S1-S6)

- **Label (target) Y**
  - Quantitative measure of disease progression
Regression

**Goal:** learn real valued mapping \( f : \mathbb{R}^d \rightarrow \mathbb{R} \)
Important choices in regression

- What types of functions $f$ should we consider? Examples

- How should we measure goodness of fit?
Example: linear regression

\[ y = f(x) \]
\[ f \text{ is linear (affine)} \]

1-dim:  \[ f(x) = ax + b \]
2-dim:  \[ f(x_1, x_2) = ax_1 + bx_2 + c \]

\[ d=\text{dim} : f(x) = w_1 x_1 + \ldots + w_d x_d + w_0 \]
\[ = \sum_{i=1}^{d} w_i x_i + w_0 \]
\[ = w^T x + w_0 \]
Homogeneous representation

\[ \mathbf{w}^T \mathbf{x} + v_0 = \mathbf{\tilde{w}}^T \mathbf{\tilde{x}} \]

where \( \mathbf{w} \in \mathbb{R}^d \) and \( \mathbf{x} \in \mathbb{R}^d \)

\[ \mathbf{\tilde{w}} = [\mathbf{w}, v_0] \]

\[ \mathbf{\tilde{x}} = [\mathbf{x}^T, 1]^T \]

\[ \Rightarrow \mathbf{w} \log \mathbf{f}(\mathbf{x}) = \mathbf{v}^T \mathbf{x} \]
Quantifying goodness of fit

\[ D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R} \]

\[ R_i = y_i - f(x_i) = y_i - w^T x_i \]

\[ \text{Cost } \mathcal{R}(w) = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (y_i - w^T x_i)^2 \]

**Note:** We’ve made 2 decisions

1) quantify error for 1 point via squared residual

2) we sum over all points
Least-squares linear regression optimization
[Legendre 1805, Gauss 1809]

- Given data set \( D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \)

- How do we find the optimal weight vector?

\[
\hat{w} = \arg \min_w \sum_{i=1}^{n} (y_i - w^T x_i)^2
\]
Method 1: Closed form solution

The problem can be solved in closed form:

\[ \hat{w} = \arg \min_w \sum_{i=1}^{n} (y_i - w^T x_i)^2 \]

Hereby:

\[ \hat{w} = (X^T X)^{-1} X^T y \]
How to solve? Example: Scikit Learn

```python
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training set
regr.fit(X_train, Y_train)

# Make predictions on the testing set
Y_pred = regr.predict(X_test)
```
Method 2: Optimization

The objective function

\[ \hat{R}(w) = \sum_{i} (y_i - w^T x_i)^2 \]

is convex!
Gradient Descent

- Start at an arbitrary \( w_0 \in \mathbb{R}^d \)
- For \( t = 1, 2, \ldots \) do
  \[
  w_{t+1} = w_t - \eta_t \nabla \hat{R}(w_t)
  \]
  Hereby, \( \eta_t \) is called learning rate
Convergence of gradient descent

- Under mild assumptions, if step size sufficiently small, gradient descent converges to a stationary point (gradient = 0)
- For convex objectives, it therefore finds the optimal solution!

In the case of the squared loss, constant stepsize $\frac{1}{2}$ converges linearly

\[ t_0 \leq t \leq T \Rightarrow \exists \alpha < 1 \text{ s.t. } (\tilde{R}(v_{t+1}) - \tilde{R}(v)) \leq \alpha (\tilde{R}(w_t) - \tilde{R}(v)) \]

\[ \Rightarrow \text{can find } \varepsilon\text{-optimal solution in } O(\ln \frac{1}{\varepsilon}) \text{ iter.} \]
Computing the gradient

\[ \nabla \hat{R}(\mathbf{w}) = \left[ \frac{\partial}{\partial w_1} \hat{R}(\mathbf{w}), \ldots, \frac{\partial}{\partial w_d} \hat{R}(\mathbf{w}) \right] \]

\[ \ln \det \nabla \hat{R}(\mathbf{w}) = \frac{d}{d\mathbf{w}} \hat{R}(\mathbf{w}) = \frac{d}{d\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 \]

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

\[ = \sum_{i=1}^{n} 2 \left( y_i - \mathbf{w} \cdot \mathbf{x}_i \right) \cdot \left(-\mathbf{x}_i\right) = -2 \sum_{i=1}^{n} \mathbf{r}_i \cdot \mathbf{x}_i \]
Demo: Gradient descent
Choosing a stepsize

What happens if we choose a poor stepsize?
Adaptive step size

- Can update the step size adaptively. For example:
- 1) Via **line search** (optimizing step size every step)
   
   \[ w_t \text{ at iter } t, \text{ have } w_t, g_t = \nabla R(v_t) \]

   Define \( \eta^*_t = \arg\min_{\eta \in [0, \infty)} R(v_t - \eta g_t) \)

- 2) „Bold driver“ heuristic
  
  - If function decreases, increase step size:
    
    \[ \text{If } R(w_{t+1}) < R(v_t) : \eta_{t+1} = \eta_t \cdot \text{acc} \]
  
  - If function increases, decrease step size:
    
    \[ \text{If } R(w_{t+1}) > R(v_t) : \eta_{t+1} = \eta_t \cdot \text{dec} \]
Demo: Gradient Descent for Linear Regression
Gradient descent vs closed form

- Why would one ever consider performing gradient descent, when it is possible to find closed form solution?

\[
\text{Closed form: } \hat{w} = (X^TX)^{-1}X^Ty
\]

\[\text{O}(n^2)\] solve (in sys. \(O(d^3)\))

\[
\text{Gradient descent: } \nabla J(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i) x_i \Rightarrow O(nd) \cdot \ln\left(\frac{1}{\epsilon}\right)
\]

- Computational complexity
- May not need an optimal solution
- Many problems don’t admit closed form solution
Other loss functions

- So far: Measure goodness of fit via squared error
- Many other loss functions possible (and sensible!)

![Graph showing various loss functions]

- Least-squares
  \[ l_2(r) = r^2 \]

- Alternatives:
  \[ l_1(r) = |r| \]
  \[ l_p(r) = |r|^p \]
  (still convex for \( p \geq 1 \))