Intro to ML: LINEAR REGRESSION

22.02.21

Instructor: Fanny Yang
Plan for today

Yesterday: Overview of different ML paradigms, examples

Focus of Part I: Supervised Learning

Focus of first weeks: Regression

Plan for today:

• Getting a feel for regression problems

• Focusing on linear regression
  
  o closed-form solution for both $n > d$ and $n < d$
  
  o Training (empirical) vs. population risk
Supervised learning pipeline

1. Labeled training set $D \in \mathcal{X} \times \mathcal{Y}$

2. Features and function class $F$

3. Training loss

4. Optimization method

Our job

ML method: Choose & train

Choose: 
```
model = sklearn.<fct_class>.<{loss,solver}>([params])
```

Train: 
```
model.fit(X_{train}, Y_{train})
```

Final Predictor $f$

Unseen samples $(x, y) \in \mathcal{X} \times \mathcal{Y}$

```
\hat{f}(x)
```

Evaluation metric comparing $\hat{f}(x)$ with $y$

```
\hat{y}_{test} = model.predict(X_{test})
```

```
\text{Evaluate: sklearn.metrics.metric}(\hat{f}(X_{test}), y_{test})
```

Model evaluation
Supervised Learning via ERM

Evaluation: Want $f(x)$ "close" to $y$ for "most" $x$

Loss per point $(x, y)$: $l(y, f(x))$

Average loss (empirical risk) for all $(x_i, y_i)$ in dataset $D$

$$\hat{R}_D(f) = \frac{1}{n} \sum_{i=1}^{n} l(y_i, f(x_i))$$

Supervised learning via Empirical Risk Minimization:

$$\hat{f}(x) = \text{argmin}_{f \in F} \hat{R}_D(f)$$
ML method: Some choices we will discuss

<table>
<thead>
<tr>
<th>Function class $F$</th>
<th>Training loss</th>
<th>Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>• linear</td>
<td>• Squared loss</td>
<td>• closed-form</td>
</tr>
<tr>
<td>• kernels</td>
<td>• SVM, logistic, cross entropy</td>
<td>• gradient descent</td>
</tr>
<tr>
<td>• trees</td>
<td>+ Penalties</td>
<td>• stochastic grad. method</td>
</tr>
<tr>
<td>• neural networks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Corresponding examples in scikit-learn:

- `linear_model`
- `kernel_ridge`
- `tree`
- `neural_network`

```python
model = sklearn.<fct_class>.(<loss>,<solver>)(<params>)
```

- `LinearRegression`, `KernelRidge`
- `LogisticRegression`
- `Ridge`, `Lasso`, `ElasticNet`

- Solver options: `'cholesky'`, `'svd'`, `'sag'`
- `SGDClassifier`, `SGDRegressor`
Regression setup

- $y$ is real number! (classification: $y$ is discrete)

- Examples:

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight route</td>
<td>Delay</td>
</tr>
<tr>
<td>Advertisement features</td>
<td>Click-through probability or sales</td>
</tr>
<tr>
<td>Real estate features</td>
<td>Sales Price</td>
</tr>
<tr>
<td>GDP, diseases, alcohol...</td>
<td>Life expectancy</td>
</tr>
</tbody>
</table>

- Running example: Diabetes
  - Input $X$: Age, gender, Body mass index, average blood pressure, blood measurements
  - Target $Y$: Quantitative measure of disease progression
What’s the best solution?

Quiz: Example for $d = 1$: Which fit is better and why?
Actually not that clear…

Answer: You think it's (B)? Well, it depends…

Training or “test” loss?

Has there been selection bias (distribution shift)?

Focus today: Minimizing training loss!
What’s the best solution?

Quiz: Example for $d = 1$: Which fit is better for the training samples and why?

Answer: Again, it depends! Is the loss sensitive to outliers? (square vs. robust loss)
ML method: Some choices we will discuss

**Function class** $F$
- linear
- kernels
- trees, kNN
- neural networks

**Training loss**
- Squared loss
- SVM, logistic, cross entropy + Penalties

**Optimization**
- closed-form
- gradient descent
- stochastic grad. method

Corresponding examples in scikit-learn:
- `linear_model`
  - `linear_model.LinearRegression`, `KernelRidge`
  - `LogisticRegression`, `Ridge`, `Lasso`, `ElasticNet`
- `kernel_ridge`
  - `SGDClassifier`, `SGDRegressor`
Linear regression: linear function + squared loss

Squared loss: \( l(y, f(x)) = (y - w^T x)^2 \)

Function class \( F_{\text{lin}} = \{ f(x) = wx + b \mid w, w_0 \in \mathbb{R} \} \)

Squared loss: \( l(y, f(x)) = (y - w^T x)^2 \)

**ERM:** \( \hat{w} = \arg\min_{f \in F_{\text{lin}}} \hat{R}_D(f) = \arg\min_{f \in F_{\text{lin}}} \frac{1}{n} \sum_{i=1}^{n} l(y_i, f(x_i)) = \arg\min_{w \in \mathbb{R}^d} \sum_i (y_i - w^T x_i)^2 \)
\(d\)-dim Linear Regression: vector notation

\[
n \hat{R}_D (w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2 = ||v||^2 \quad \text{with} \quad v_i = y_i - x_i^T w
\]

\[
v = y - Xw \quad \text{with} \quad X \in \mathbb{R}^{n \times d} \quad \text{and} \quad X = \begin{pmatrix} -x_1 \\
\vdots \\
-x_n \end{pmatrix} = (\bar{x}_1, \ldots, \bar{x}_d)
\]

\[
\hat{w} = \arg\min_{w \in \mathbb{R}^d} \hat{R}_D (w) = \arg\min_{w \in \mathbb{R}^d} ||y - Xw||^2 = \arg\min_{w \in \mathbb{R}^d} ||y - Xw||
\]

\[
\begin{array}{c}
\begin{pmatrix} n \\
\vdots \\
n \end{pmatrix} \quad \approx \quad \begin{pmatrix} d \\
\vdots \\
d \end{pmatrix} \\
\begin{pmatrix} y \\
\vdots \\
y \end{pmatrix} \quad \approx \quad \begin{pmatrix} X \\
\vdots \\
X \end{pmatrix}
\end{array}
\]

Case 1: \(d < n \rightarrow \) one best solution

Case 2: \(d > n \rightarrow \) many exact solutions!
Case 1: optimal solution for $d < n$

How does the closed form solution $\hat{w}$ look like as a function of $y$, $X$?

$\hat{w} = \arg\min_w \| y - Xw \|

Example in $d = 2, n = 3$:

- Closest point to $y$ on $\text{span}(X)$ is $\Pi_X y$
- $\Pi_X y$ is orthogonal projection of $y$ onto $\text{span}(X)$

$\hat{y} = X\hat{w} = \sum_{i=1}^{d} \hat{w}_i \tilde{x}_i$

will be in subspace spanned by columns of $X$ ($\text{span} X$)

$R^n$$\rightarrow$$R^{d-1}$

$X = \begin{pmatrix} x_1 & \cdots & x_d \end{pmatrix}$

$\hat{y} = \Pi_X y$

has smallest $\| y - \hat{y} \|$
Orthogonal projection via normal equations

Closed-form for projection of \( y \in \mathbb{R}^n \) onto \( \text{span}(X) \) (subspace spanned by columns of \( X \))

- \( \Pi_X y \) in \( \text{span}(X) \) \iff \( \Pi_X y = X \hat{w} \) for some \( \hat{w} \)

- \( X \hat{w} \) is orthogonal projection \iff residual \( y - X\hat{w} \) orthogonal to all \( v \in \text{span}(X) \)
  \iff \( (y - X\hat{w})^T X \beta = 0 \) for all \( \beta \in \mathbb{R}^d \)

- Hence we require \( X^T (y - X\hat{w}) = 0 \) \iff \( X^T y = X^T X \hat{w} \) (normal equations!)
Case 1: closed form solution for $\Pi_X y$

Remember $X \in \mathbb{R}^{n \times d}$ and $d < n$. When is this equation solvable? Select all that apply

(A) only when $\exists \hat{w}$ s.t. $y = X\hat{w}$  
(B) only when $X^T X$ invertible  
(C) only when $\text{rank}(X) = d$  
(D) always

Answer: Always possible!

- If invertible, $\hat{w} = (X^T X)^{-1}X^T y$ and hence, projection $\Pi_X y = X(X^T X)^{-1}X^T y$

- What if $\text{rank}(X) < d$ and hence $X^T X$ not invertible?

  Same formula, but replacing $(X^T X)^{-1}$ by pseudoinverse $(X^T X)^\dagger$ (by def. satisfying $A^\dagger A A^\dagger = A^\dagger$)

- Compute using inverting non-zero singular values (see math recap)

  Most generally $\hat{w} = (X^T X)^\dagger X^T y$ and $\Pi_X y = X\hat{w} = X(X^T X)^\dagger X^T y$
What should an optimum point satisfy

Coming from $d = 1$: Necessary conditions

- Minimum is always a stationary point (for differentiable functions $g: \nabla g(w) = 0$)
- Stationary points are optimal points for quadratic functions with psd matrix ($\rightarrow$ convexity (later))
- $\nabla_w \hat{R}_D (w) = \nabla_w ||y - Xw||^2 = X^T (Xw - y) = 0 \rightarrow$ normal equations! $\rightarrow$ satisfied by $\hat{w} = (X^T X)^{-1} X^T y$
What happens when $n < d$?

Simple model: fix some $d = 40$, start with $n = 400$. What happens when we decrease $n$?

$d < n$: Underparameterized (unique solution)

$d > n$: Overparameterized (many solutions achieve $y = Xw$ exactly)

Figure: Model details are in demo (with code), will be uploaded to website
What happens when $n < d$?

Simple model: fix some $d = 40$, start with $n = 400$. What happens when we decrease $n$?

$d < n$: Underparameterized (unique solution)

$d > n$: Overparameterized (many solutions achieve $y = Xw$ exactly)

Figure: Model details are in demo (with code), will be uploaded to website
Case 2: min norm solution $d > n \geq \text{rank}(X)$

- Overparameterized $\rightarrow$ many solutions that achieve $y = Xw$ exactly. Which do we find?
- scikit learn and GD find minimum norm solution: $\min ||w|| \text{ s.t. } y = Xw$

What is the closed form solution?

- Simple argument shows that suffices to search for $w \in \text{span}(X^T)$ (see final slides & tutorial)
- Since $w \in \text{span}(X^T)$, $\rightarrow w = X^T v$ for some $v \in \mathbb{R}^n$ $\rightarrow \hat{w} = X^T (XX^T)^\dagger y$
- Using SVD, can check that $\hat{w} = (X^TX)^\dagger X^T y$

Linear regression estimate $\hat{w} = (X^TX)^\dagger X^T y$ for any $n, d$
How good is our linear regression estimate?
ML pipeline for fixed model

True distribution $(x, y) \sim P_{xy}$

Labeled training set $D = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$

Our job

ML method: Choose & train

1. Features and function class $F$
2. Training loss
3. Optimization method

Model evaluation

Unseen samples $(x, y) \in \mathcal{X} \times \mathcal{Y}$

Final Predictor $\hat{f}$

Evaluation metric comparing $\hat{f}(x)$ with $y$
From empirical to population risk

- When would there be a chance that $\hat{f}(x)$ performs well on new test samples?

  If $(x_i, y_i)$ in training set are “similar” to test samples $(x, y)$

- In statistical learning, we model similarity by the i.i.d. assumption $(x_i, y_i) \sim P_{xy}$ and $(x, y) \sim P_{xy}$

- Common evaluation of a function $f$: population risk of $f$ on $P_{xy}$ is expected test loss with $(x, y) \sim P_{xy}$

  $$R(f) = \mathbb{E}_{(x,y) \sim P_{xy}} l(y, f(x))$$

- Remember $\hat{f}$ minimizes the empirical risk (expected point-loss over empirical distribution)

  $$\hat{R}_D(f) = \frac{1}{n} \sum_i l(y_i, f(x_i))$$
How good of a proxy is $\hat{R}_D(f)$ for $R(f)$

If we know $\hat{R}_f$ (training loss) is small, can we deduce expected test loss is also small?

- $f^*$ be minimizer of $R(f)$
- At $\hat{f}$, empirical risk systematically underestimates population risk
  $\mathbb{E}_D \hat{R}_D(\hat{f}) \leq R(\hat{f})$
Underestimation of population risk

\[ \mathbb{E}_D \hat{R}_D(\hat{f}_D) = \mathbb{E}_D \min_{f \in F} \hat{R}_D(f) \]

For any \( f' \in F \)

\[ \mathbb{E}_D \min_{f \in F} \hat{R}_D(f) \leq \mathbb{E}_D \hat{R}_D(f') \]

Linearity of expectation

\[ \leq \min_{f \in F} \mathbb{E}_D \hat{R}_D(f) \]

\[ = \min_{f \in F} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_D l(y_i, f(x_i)) \]

\[ = \min_{f \in F} R(f) = R(f^*) \leq R(\hat{f}_D) \]

for all draws of training set \( D \)
Simple example

Assume simplest setting:

- 1-d linear regression: \( F = \{ f(x) = wx \mid w \in \mathbb{R} \} \)
- Input distribution: \( x_i \sim N(0,1) \)
- Noisy observations: \( y_i = w^*x_i + \epsilon_i \) with \( \epsilon_i \sim N(0,\sigma^2) \)

\[ \Rightarrow \text{Population risk: } R(w) = \mathbb{E}_{(x,y) \sim p_{xy}} (y - wx)^2 = (w^* - w)^2 + \sigma^2 \]

and empirical risk: \( \hat{R}_D(w) = \frac{1}{n} \sum_{i=1}^{n} ((w^* - w)x_i + \epsilon_i)^2 \)

Note: Both are quadratics in \( w \)!
Why is there hope we’re learning the right thing?

**Demo**

Law of large numbers: Empirical mean converges to expectation as $n \to \infty$

That is pointwise: $\hat{R}_D(f) = \frac{1}{n} \sum_i l(y_i, f(x_i)) \to R(f) = \mathbb{E}_{(x,y) \sim P_{xy}} l(y, f(x))$

What’s your best guess: Does this imply (i) $\hat{f} \to f^*$? How about (ii) $\hat{R}_D(f) \to R(\hat{f})$?

(A) Yes to (i)  (B) Yes to (ii)  (C) Yes to both  (D) No to both

Answer: At least the same technique does not work as $\hat{R}_D(f)$

(keyword: uniform convergence $\to$ Guarantees for Machine Learning)
Summary and outlook

• Closed form minimizer of square loss with linear model → linear regression estimate
• Training error is not enough to evaluate how good a predictor is
• How test loss changes with ratio $\frac{d}{n}$ for simple regression model

Next lecture:
• Population risk of linear regression estimate
• Regularization and bias-variance tradeoff