Machine Learning for Networks: Regression (continued) and Classification

Andrea Araldo

September 14, 2023

Regression (continued)

- Polynomial Regression
- Variance vs. Bias Trade-Off
- Regularization
- Scaling
- Feature Selection

Classification

- Logistic Regression
- Classification Performance
- Class imbalance

Section 1

Polynomial Regression and hyper-parameter tuning

Univariate Polynomial Regressions

- A univariate polynomial regression of degree p is

$$h_\theta(x) = \theta_0 + \theta_1 \cdot x + \theta_2 \cdot x^2 + \ldots \theta_p \cdot x^p$$

- $p = 1$: linear
  $p = 2$: quadratic
  $p = 3$: cubic
  …
- Equivalent to linear regression with features

$$x, x^2, \ldots, x^p$$

- p is a *hyper-parameter*: parameter of the learning algorithm.
- How to choose p?

Multi-variate polynomial regression

- With $j = 1 \dots N$ features, all terms of degree 2 are included:

$$
\begin{aligned}
h_\theta(\mathbf{x}) = \theta_0 + \theta_1 \cdot x_1 + \dots + \theta_N \cdot x_N \\
+ \theta_{N+1} \cdot x_1^2 + \dots + \theta_{N+N} \cdot x_N^2 \\
+ \theta_{1,2} \cdot x_1 x_2 + \theta_{1,3} \cdot x_1 x_3 + \dots + \theta_{1,N} \cdot x_1 x_N \\
+ \theta_{2,3} \cdot x_2 x_3 + \theta_{2,4} \cdot x_2 x_4 + \dots
\end{aligned}
$$

- A pol. regression of degree p includes the following terms:
  - Bias term

$$\theta_0$$

  - Powers of features

$$x_j^k \qquad k = 1, \dots, p$$

  - Mixed terms of power 2:

$$x_j \cdot x_{j'} \qquad j' > j$$

  - Mixed terms of power 3

$$x_j \cdot x_{j'} \cdot x_{j''} \qquad j'' > j' > j$$

  - ...
  - Mixed terms of power p

Hyper-parameter tuning

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	720
3.638107	0.005493	true	q240p	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	480
4.940168	0.008851	true	q1080p	1080
4.940168	0.008851	true	q1080p	1080
9.239532	0.016335	true	q720p	720

Hyper-parameter tuning

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	720
3.638107	0.005493	true	q240p	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	480
4.940168	0.008851	true	q1080p	1080
4.940168	0.008851	true	q1080p	1080
9.239532	0.016335	true	q720p	720

TRAINING SET

TEST SET

- Divide training and test sets

Hyper-parameter tuning

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	720
3.638107	0.005493	true	q240p	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	480
~~4.940168~~	~~0.008851~~	~~true~~	~~q1080p~~	~~1080~~
~~4.940168~~	~~0.008851~~	~~true~~	~~q1080p~~	~~1080~~
~~9.239532~~	~~0.016335~~	~~true~~	~~q720p~~	~~720~~

TRAINING SET

TEST SET

- Divide training and test sets
- Use only training set

Hyper-parameter tuning

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	720
3.638107	0.005493	true	q240p	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	480
4.940168	0.008851	true	q1080p	1080
4.940168	0.008851	true	q1080p	1080
9.239532	0.016335	true	q720p	720

TRAINING SET

TEST SET

- Divide training and test sets
- Use only training set
- For all the hyper-parameter values
  - Construct a model with such values
  - Compute cross-validation error

Hyper-parameter tuning

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	720
3.638107	0.005493	true	q240p	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	480
4.940168	0.008851	true	q1080p	1080
4.940168	0.008851	true	q1080p	1080
9.239532	0.016335	true	q720p	720

TRAINING SET

TEST SET

- Divide training and test sets
- Use only training set
- For all the hyper-parameter values
  - Construct a model with such values
  - Compute cross-validation error
- Select the model with the smallest cross-validation error

Hyper-parameter tuning

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	720
3.638107	0.005493	true	q240p	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	480
4.940168	0.008851	true	q1080p	1080
4.940168	0.008851	true	q1080p	1080
9.239532	0.016335	true	q720p	720

TRAINING SET

TEST SET

- Divide training and test sets
- Use only training set
- For all the hyper-parameter values
  - Construct a model with such values
  - Compute cross-validation error
- Select the model with the smallest cross-validation error
- Train the selected model on the training set
- Test error on the test set

Hyper-parameter tuning

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	720
3.638107	0.005493	true	q240p	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	480
4.940168	0.008851	true	q1080p	1080
4.940168	0.008851	true	q1080p	1080
9.239532	0.016335	true	q720p	720

TRAINING SET

TEST SET

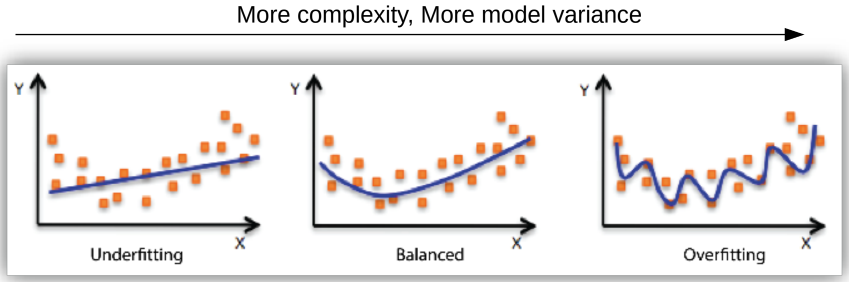
- Divide training and test sets
- Use only training set
- For all the hyper-parameter values
  - Construct a model with such values
  - Compute cross-validation error
- Select the model with the smallest cross-validation error
- Train the selected model on the training set
- Test error on the test set

We have only used the training set to select the best parameter

Let's code ...

Go to notebook
03.regression_contd-and-classification/a.polynomial-regression.ipynb

Complexity and Variance

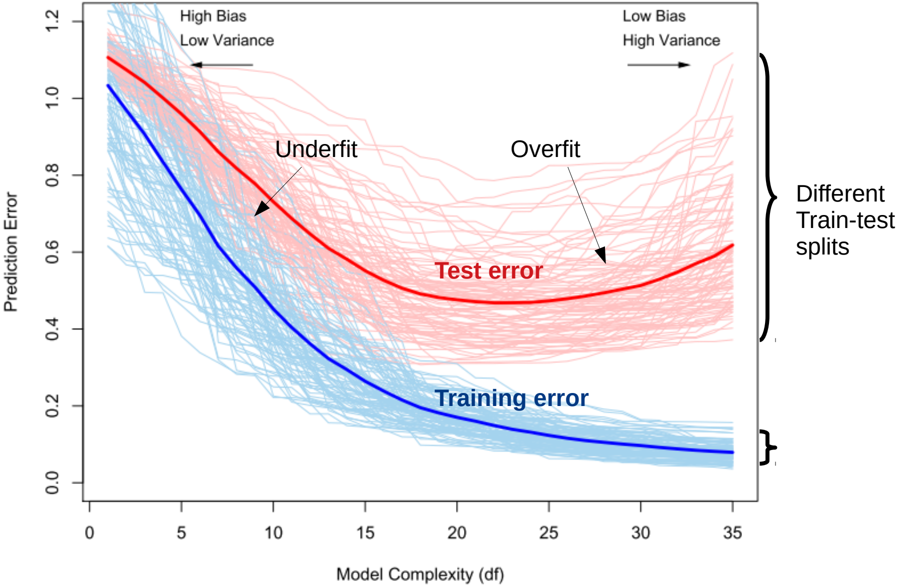


Example of polynomial regression with degree 1 (linear), and then higher degrees

Image from [AWS].

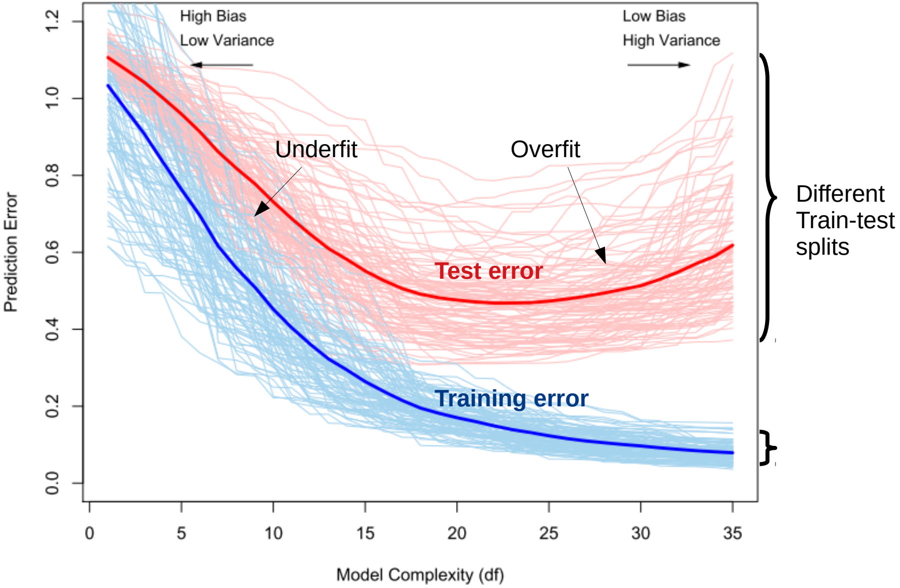
Complexity and Variance

Higher $p \implies$ higher complexity $\implies$ higher variance
(the model adapts too flexibly to the training data)



Bias-Variance trade-off

If you reduce bias (on the training set) you increase the variance. And vice-versa.
This is a fundamental limit of Machine Learning [KW96].



Section 2

Regularization

Regularization

- Force the model to be simple.
  Cost function:

$$J(\theta, \mathbf{X}, \hat{\mathbf{y}}) = \frac{1}{M}\sum_{i=1}^{M}\left(y^{(i)} - h_\theta(\mathbf{x}^{(i)})\right)^2 \underbrace{+\alpha\sum_{j=1}^{N}\theta_j^2}_{\text{regularization term}}$$

- Parameters forced to be small $\Longrightarrow$ less overfit
- Bias term θ_0 not regularized. Why?

Regularization

- Force the model to be simple.
  Cost function:

$$J(\theta, \mathbf{X}, \hat{\mathbf{y}}) = \frac{1}{M} \sum_{i=1}^{M} \left(y^{(i)} - h_\theta(\mathbf{x}^{(i)}) \right)^2 \quad \underbrace{+\alpha \sum_{j=1}^{N} \theta_j^{\,2}}_{\text{regularization term}}$$

- Parameters forced to be small $\implies$ less overfit
- Bias term θ_0 not regularized. Why?
  - It is just an offset. It does not add complexity.

Regularization

- Force the model to be simple.
  Cost function:

$$J(\theta, \mathbf{X}, \hat{\mathbf{y}}) = \frac{1}{M}\sum_{i=1}^{M}\left(y^{(i)} - h_\theta(\mathbf{x}^{(i)})\right)^2 \quad \underbrace{+\alpha\sum_{j=1}^{N}\theta_j^{\,2}}_{\text{regularization term}}$$

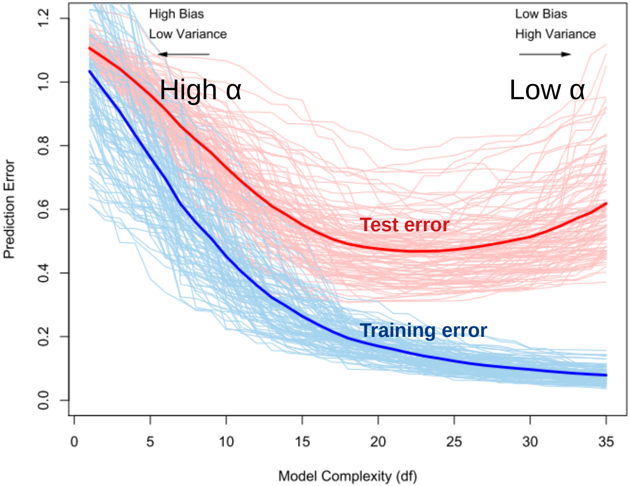
- Parameters forced to be small $\Longrightarrow$ less overfit
- Bias term θ_0 not regularized. Why?
  - It is just an offset. It does not add complexity.
- Should regularization term considered when evaluating test error?

$$J_{\text{train}} = \frac{1}{\left|\mathscr{D}^{\text{train}}\right|}\sum_{i\in\mathscr{D}^{\text{train}}}\left(y^{(i)} - h_\theta(\mathbf{x}^{(i)})\right)^2 \quad \underbrace{+\alpha\sum_{j=1}^{N}\theta_j^{\,2}}_{\text{regularization term}}$$

$$J_{\text{test}} = \frac{1}{\left|\mathscr{D}^{\text{test}}\right|}\sum_{i\in\mathscr{D}^{\text{test}}}\left(y^{(i)} - h_\theta(\mathbf{x}^{(i)})\right)^2$$

Effects of α

$$J_{\text{train}} = \frac{1}{|\mathscr{D}^{\text{train}}|} \sum_{i \in \mathscr{D}^{\text{train}}} \left(y^{(i)} - h_{\theta}(\mathbf{x}^{(i)})\right)^2 \underbrace{+\alpha \sum_{j=1}^{N} \theta_j^{\,2}}_{\text{regularization term}}$$



- What if $\alpha \to 0$?

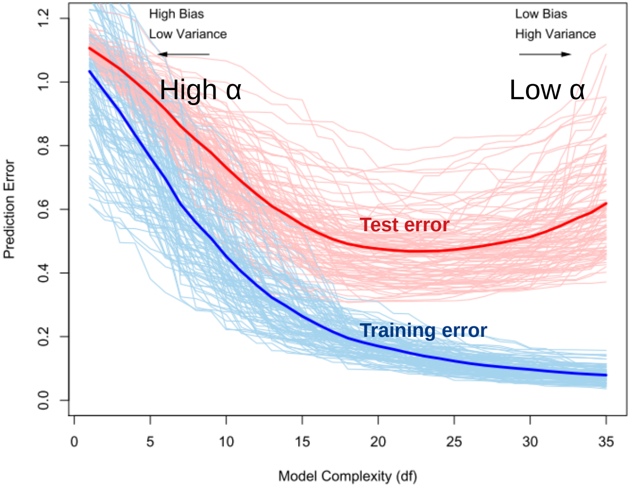
  Linear regression

- And if $\alpha \to +\infty$?

  Only θ_0

Effects of α

$$J_{\text{train}} = \frac{1}{|\mathcal{D}^{\text{train}}|} \sum_{i \in \mathcal{D}^{\text{train}}} \left(y^{(i)} - h_\theta(\mathbf{x}^{(i)})\right)^2 \underbrace{+\alpha \sum_{j=1}^{N} \theta_j^2}_{\text{regularization term}}$$



- What if $\alpha \to 0$?

  Linear regression

- And if $\alpha \to +\infty$?

  Only θ_0

- Suppose
  - you try different α and
  - the best error is with $\alpha \to +\infty$.

  What do you conclude?

  In this case, the best model is the simple average of y.

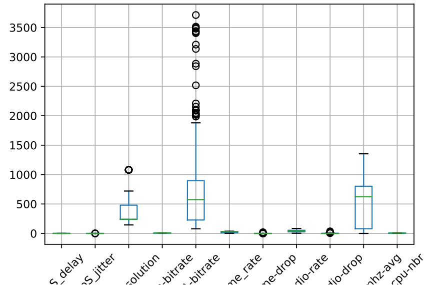
Let’s code ...

Section 3

Scaling

Regularization and scaling

Features may have different magnitudes



- Regularization squashes blindly all features uniformly.
- "Small" features would need instead larger
- Need to scale features before applying regularization.

$$J(\theta, \mathbf{X}, \mathbf{y}) = \frac{1}{M} \sum_{i=1}^{M} \left(y^{(i)} - h_\theta(\mathbf{x}^{(i)}) \right)^2 \quad \underbrace{+\alpha \sum_{j=1}^{N} \theta_j^{\,2}}_{\text{regularization term}}$$

$$\theta^* = \arg\min_\theta J(\theta, \mathbf{X}, \mathbf{y})$$

Standard scaler

- If μ_j is avg of j-feature and σ_j the stdev:
  Standard Scaler

$$x_j^{(i)\prime} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

Which is the correct way of applying scaling?

- Divide $(\mathbf{X}, \mathbf{y}) \to (\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}), (\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$
- $\mathbf{X}_{\text{train}}{}' = \text{scale}(\mathbf{X}_{\text{train}})$
- $\mathbf{X}_{\text{test}}{}' = \text{scale}(\mathbf{X}_{\text{test}})$
- Train the model using $(\mathbf{X}_{\text{train}}{}', \mathbf{y}_{\text{train}})$
- Test using $(\mathbf{X}_{\text{test}}{}', \mathbf{y}_{\text{test}})$ using $\mu_j, \sigma_j, \min_j, \max_j$ found in training

vs.

- $\mathbf{X}' = \text{scale}(\mathbf{X})$
- Divide $(\mathbf{X}', \mathbf{y}) \to (\mathbf{X}_{\text{train}}{}', \mathbf{y}_{\text{train}}), (\mathbf{X}_{\text{test}}{}', \mathbf{y}_{\text{test}})$
- Train the model using $(\mathbf{X}_{\text{train}}{}', \mathbf{y}_{\text{train}})$
- Test using $(\mathbf{X}_{\text{test}}{}', \mathbf{y}_{\text{test}})$

Standard scaler

- If μ_j is avg of j-feature and σ_j the stdev:

Standard Scaler

$$x_j^{(i)\prime} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

Which is the correct way of applying scaling?

- Divide $(\mathbf{X}, \mathbf{y}) \rightarrow (\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}), (\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$
- $\mathbf{X}_{\text{train}}{}' = \text{scale}(\mathbf{X}_{\text{train}})$
- $\mathbf{X}_{\text{test}}{}' = \text{scale}(\mathbf{X}_{\text{test}})$
- Train the model using $(\mathbf{X}_{\text{train}}{}', \mathbf{y}_{\text{train}})$
- Test using $(\mathbf{X}_{\text{test}}{}', \mathbf{y}_{\text{test}})$ using $\mu_j, \sigma_j, \min_j, \max_j$ found in training

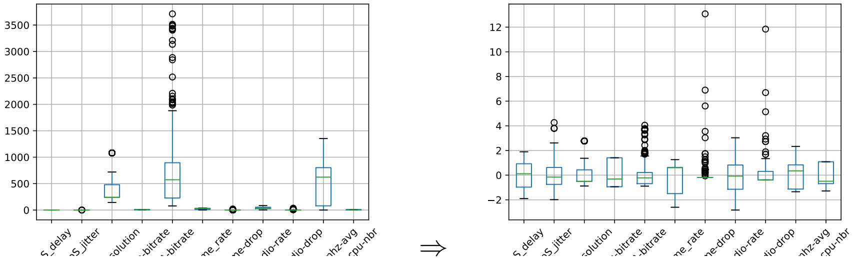
vs.

- $\mathbf{X}' = \text{scale}(\mathbf{X})$
- Divide $(\mathbf{X}', \mathbf{y}) \rightarrow (\mathbf{X}_{\text{train}}{}', \mathbf{y}_{\text{train}}), (\mathbf{X}_{\text{test}}{}', \mathbf{y}_{\text{test}})$
- Train the model using $(\mathbf{X}_{\text{train}}{}', \mathbf{y}_{\text{train}})$
- Test using $(\mathbf{X}_{\text{test}}{}', \mathbf{y}_{\text{test}})$

Data Leakage (Ch.8 of [Teo19])

In the 2nd case we would calculate $\mu_j, \sigma_j, \min_j, \max_j$ using data from test set

Effect of scaling



When is scaling important

Is scaling important in a linear regression?

When is scaling important

Is scaling important in a linear regression?

- It does not affect the accuracy of the model
  - Because coefficients can scale based on the feature magnitude.
- But it's good for interpretability, **when features are standardized**
  - Since we impose the stddev of all features to be 1, the value of the coefficient is an indication of **feature importance**
    - (how much a variation of a feature impacts the target)

Is scaling important in a polynomial regression?

Section 4

Feature selection

Feature selection

- We have already seen some methods:
  - Check the Pearson's correlation
  - Run a lin.regr. on the scaled dataset and check the magnitude of the coefficients.
  - See if a model improves/deteriorates when removing a feature
- Another method: **Recursive Feature Elimination** (RFE)
  - Standardize your features
  - Train your model with all features
  - Remove the feature with the smallest coeff
  - Train the model again
  - Remove the feature with the smallest coeff
  - ...
  - Repeat until you are left with N features.
- Why do we need to standardize the features?

  Otherwise the coefficient weights are not an indication of feature importance.
- RFE + Cross Validation (RFECV)
  - Repeat the process for different N and select the N providing the smallest cross-validation error.

Let's code ...

Go to notebook 03.regression_contd-and-classification/b.regularization

Section 5

Classification: Logistic Regression

Classification

Supervised ML task where the labels are in a finite set.

- Ex.: Classify video resolution based on network information
  Labels are 144p, 360p, etc.

Binomial Logistic Regression

- Classes $k = 0$ (negative) and 1 (positive).
- We do not predict directly the class k of sample $\mathbf{x}$
- We instead predict probabilities
  - The predicted probability of being positive is

$$\hat{p}_1^{(i)} = \mathbb{P}\left[\mathbf{x}^{(i)} \text{ is of class } 1\right]$$
$$= h_\theta(\mathbf{x}^{(i)}) = \sigma(\theta^T \cdot \mathbf{x}^{(i)})$$

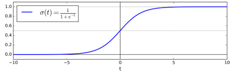
  - The predicted probability of being negative is

$$\hat{p}_0^{(i)} = 1 - \hat{p}_1^{(i)}$$

  - If $y^{(i)}$ is the true class of sample $\mathbf{x}^{(i)}$, the predicted probability of being of the true class is

$$\hat{p}_{y^{(i)}}^{(i)}$$

- *Sigmoid* $\sigma(t) = \frac{1}{1+e^{-t}}$



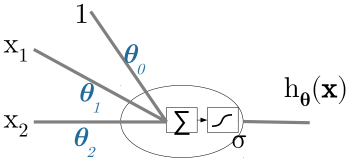
Picture from [Gér17]

- The predicted label is:

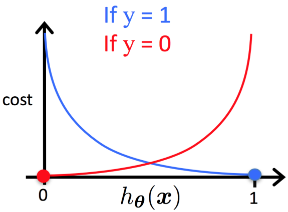
$$\hat{y}^{(i)} = \begin{cases} 1 & \text{if } \hat{p}_1^{(i)} \geq 0.5 \\ 0 & \text{if } \hat{p}_1^{(i)} < 0.5 \end{cases}$$

Neural Network

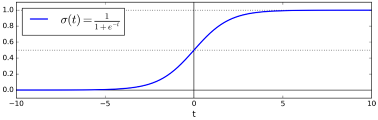
Logistic regression is a NN with one neuron



Log-Loss function



Picture from stackexchange



- How can we find:

$$\theta^* = \arg\min_{\theta} J(\theta, \mathbf{X}, \mathbf{y})?$$

- For any sample $(\mathbf{x}^{(i)}, y^{(i)})$:

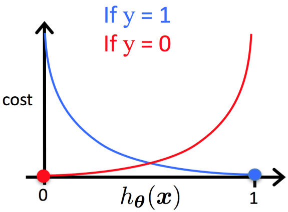
$$J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \triangleq -\ln\left(\hat{p}^{(i)}_{y^{(i)}}\right) = \begin{cases} -\ln\left(\hat{p}^{(i)}_1\right) & \text{if } y^{(i)} = 1 \\ -\ln\left(\hat{p}^{(i)}_0\right) & \text{if } y^{(i)} = 0 \end{cases}$$

$$= \begin{cases} -\ln\left(\sigma(\theta^T \cdot \mathbf{x}^{(i)})\right) & \text{if } y^{(i)} = 1 \\ -\ln\left(1 - \sigma(\theta^T \cdot \mathbf{x}^{(i)})\right) & \text{if } y^{(i)} = 0 \end{cases}$$

- For the entire dataset $(\mathbf{X}, \mathbf{y})$:

$$J(\theta, \mathbf{X}, \mathbf{y}) \triangleq \frac{1}{M} \sum_{i=1}^{M} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \tag{1}$$

Log-Loss function



Picture from stackexchange

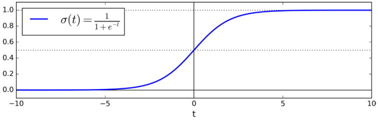
- For any sample $(\mathbf{x}^{(i)}, y^{(i)})$:

$$J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \triangleq -\ln\left(\hat{p}^{(i)}_{y^{(i)}}\right) = \begin{cases} -\ln\left(\hat{p}^{(i)}_1\right) & \text{if } y^{(i)} = 1 \\ -\ln\left(\hat{p}^{(i)}_0\right) & \text{if } y^{(i)} = 0 \end{cases}$$

$$= \begin{cases} -\ln\left(\sigma(\theta^T \cdot \mathbf{x}^{(i)})\right) & \text{if } y^{(i)} = 1 \\ -\ln\left(1 - \sigma(\theta^T \cdot \mathbf{x}^{(i)})\right) & \text{if } y^{(i)} = 0 \end{cases}$$

- For the entire dataset $(\mathbf{X}, \mathbf{y})$:

$$J(\theta, \mathbf{X}, \mathbf{y}) \triangleq \frac{1}{M} \sum_{i=1}^{M} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \tag{1}$$



- How can we find:

$$\theta^* = \arg\min_{\theta} J(\theta, \mathbf{X}, \mathbf{y})?$$

- For any $(\mathbf{x}^{(i)}, y^{(i)})$, the loss function is derivable and convex:

$$\nabla J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = \begin{cases} \nabla\left[-\ln\left(\sigma(\theta^T \cdot \mathbf{x}^{(i)})\right)\right] & \text{if } y^{(i)} = 1 \\ \nabla\left[-\ln\left(1 - \sigma(\theta^T \cdot \mathbf{x}^{(i)})\right)\right] & \text{if } y^{(i)} = 0 \end{cases}$$

- $\Longrightarrow J(\theta, \mathbf{X}, \mathbf{y})$ is also derivable and convex
- $\Longrightarrow$ We can use gradient descent.

Gradient Descent

- At each iteration

$$\theta := \theta - \eta \cdot \nabla_\theta J(\theta, \mathbf{X}, \mathbf{y})$$

where (see (1))

$$\nabla_\theta J(\theta, \mathbf{X}, \mathbf{y}) = \frac{1}{M}\sum_{i=1}^{M} \nabla_\theta J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = \frac{1}{M}\sum_{i=1}^{M} \begin{bmatrix} \frac{\partial}{\partial\theta_0} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \\ \vdots \frac{\partial}{\partial\theta_N} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \end{bmatrix}$$

- For any sample i we can compute that[1] (No need to learn it by heart):

$$\frac{\partial}{\partial\theta_j} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = \left(\underbrace{\underbrace{\sigma(\theta^T \cdot \mathbf{x}^{(i)})}_{} - y^{(i)}}_{} \right) \cdot x_j^{(i)}$$

[1] Eq. 4.18 of [Gér17]

- At each iteration

$$\theta := \theta - \eta \cdot \nabla_\theta J(\theta, \mathbf{X}, \mathbf{y})$$

where (see (1))

$$\nabla_\theta J(\theta, \mathbf{X}, \mathbf{y}) = \frac{1}{M}\sum_{i=1}^{M} \nabla_\theta J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = \frac{1}{M}\sum_{i=1}^{M} \begin{bmatrix} \frac{\partial}{\partial\theta_0} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \\ \vdots \frac{\partial}{\partial\theta_N} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \end{bmatrix}$$

- For any sample i we can compute that[1] (No need to learn it by heart):

$$\frac{\partial}{\partial\theta_j} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = \left(\underbrace{\underbrace{\sigma(\theta^T \cdot \mathbf{x}^{(i)})}_{\hat{p}^{(i)}} - y^{(i)}}_{} \right) \cdot x_j^{(i)}$$

[1]Eq. 4.18 of [Gér17]

Gradient Descent

- At each iteration

$$\theta := \theta - \eta \cdot \nabla_\theta J(\theta, \mathbf{X}, \mathbf{y})$$

where (see (1))

$$\nabla_\theta J(\theta, \mathbf{X}, \mathbf{y}) = \frac{1}{M} \sum_{i=1}^{M} \nabla_\theta J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = \frac{1}{M} \sum_{i=1}^{M} \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \\ \vdots \frac{\partial}{\partial \theta_N} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \end{bmatrix}$$

- For any sample i we can compute that[1] (No need to learn it by heart):

$$\frac{\partial}{\partial \theta_j} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = \left(\underbrace{\underbrace{\sigma(\theta^T \cdot \mathbf{x}^{(i)})}_{\hat{p}^{(i)}} - y^{(i)}}_{-\varepsilon^{(i)}} \right) \cdot x_j^{(i)}$$

[1] Eq. 4.18 of [Gér17]

Gradient Descent

- At each iteration

$$\theta := \theta - \eta \cdot \nabla_\theta J(\theta, \mathbf{X}, \mathbf{y})$$

where (see (1))

$$\nabla_\theta J(\theta, \mathbf{X}, \mathbf{y}) = \frac{1}{M}\sum_{i=1}^{M} \nabla_\theta J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = \frac{1}{M}\sum_{i=1}^{M} \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \\ \vdots \frac{\partial}{\partial \theta_N} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) \end{bmatrix}$$

- For any sample i we can compute that[1] (No need to learn it by heart):

$$\frac{\partial}{\partial \theta_j} J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = \left(\underbrace{\underbrace{\sigma(\theta^T \cdot \mathbf{x}^{(i)})}_{\hat{p}^{(i)}} - y^{(i)}}_{-\varepsilon^{(i)}} \right) \cdot x_j^{(i)}$$

- Therefore

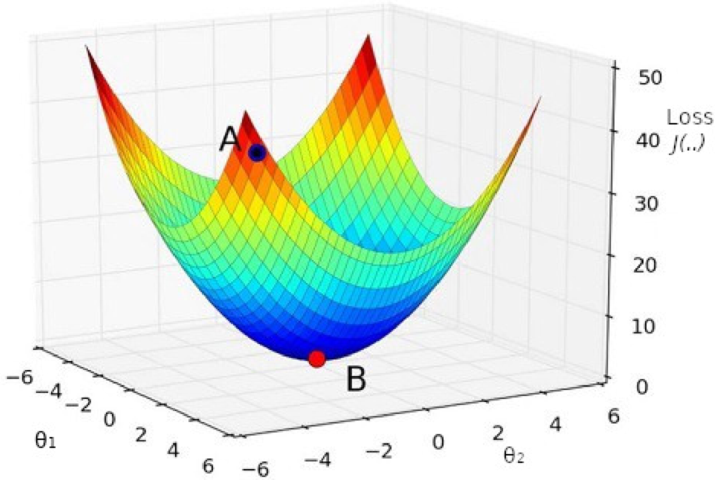
$$\nabla_\theta J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = -\varepsilon^{(i)} \cdot \mathbf{x}^{(i)}$$

[1] Eq. 4.18 of [Gér17]

Gradient Descent

At each iteration

$$\theta := \theta - \eta \cdot \nabla_\theta J(\theta, \mathbf{X}, \mathbf{y})$$



Training with Gradient Descent

1. **Full** Gradient Descent:
   - Initialize a random θ
   - Compute $h_\theta(\mathbf{x}^{(i)})$ for all $(\mathbf{x}^{(i)}, y^{(i)}) \in \mathscr{D}^{\text{train}}$
   - Compute the residuals

$$\varepsilon^{(i)} = h_\theta(\mathbf{x}) - y^{(i)}$$

   - Apply the update:

$$\theta := \theta - \eta \cdot \underbrace{\frac{1}{M}\sum_{i=1}^{M} \varepsilon^{(i)} \cdot \mathbf{x}^{(i)}}_{\nabla_\theta J(\theta, \mathbf{X}, \mathbf{y})}$$

   - Repeat several **epochs**.

The more the error on a $\mathbf{x}^{(i)}$, the more its contribution to the update.

Problem: what happens if $\mathscr{D}^{\text{train}}$ is huge?

2. **Stochastic** gradient descent
   - Randomly select one sample $(\mathbf{x}^{(i)}, y^{(i)})$
   - Directly update

$$\theta := \theta - \eta \cdot \underbrace{\varepsilon^{(i)} \cdot \mathbf{x}^{(i)} \nabla}_{\nabla_\theta J(\theta, \mathbf{x}^{(i)}, y^{(i)})}$$

   - Why *stochastic*:
     we apply a quantity which on expectation is equal to the actual gradient
3. **Batch** gradient descent
   - Partition $\mathscr{D}^{\text{train}}$ into batches
   - For each batch
     – Predict all the data

Logistic Regression is a linear classifier

Decision boundary: Surface of $\mathbb{R}^{N+1}$ that divides the region in which the classifier predicts 1 and the region in which it predicts 0.

Theorem

The decision boundary of Logistic Regression is a hyperplane

Logistic regression predicts 1 if

$$\hat{p}_1^{(i)} = h_\theta(\mathbf{x}) = \sigma(\theta^T \cdot \mathbf{x}^{(i)}) \geq 0.5$$

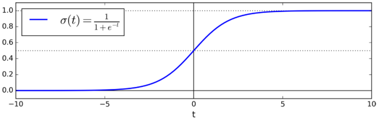
$$\iff$$

$$\theta^T \cdot \mathbf{x}^{(i)} \geq 0$$

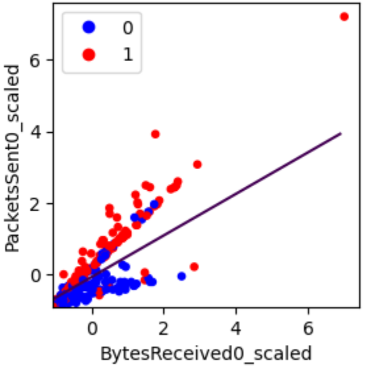
Therefore, the boundary decision is the set of $\mathbf{x}$ such that

$$\theta^T \cdot \mathbf{x} = 0$$

This surface is described by a linear equation, and thus it is a hyperplane.



Picture above from [Gér17]



Multinomial Logistic Regression

Extension to multiple classes.

- Each class has its weight parameter $\theta_k \in \mathbb{R}^{N+1}$, except the last
- Compute a *score* $s_k(\mathbf{x}) \triangleq {\theta_k}^T \cdot \mathbf{x}$
- For any $\mathbf{x}$, we have the score of all classes

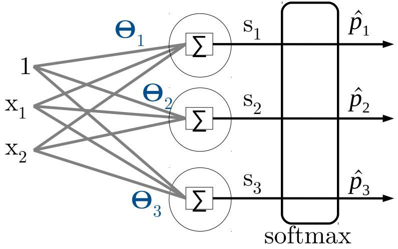
$$s_1(\mathbf{x}), \ldots, s_K(\mathbf{x})$$

- We define that:

$$\hat{p}_k = \mathbb{P}\left[\mathbf{x} \in \text{class } k\right] = \text{softmax}(s_k(\mathbf{x}))$$
$$\triangleq \frac{\exp s_k(\mathbf{x})}{\sum_{z=1}^{K} \exp s_z(\mathbf{x})} = \frac{\exp\left({\theta_k}^T \cdot \mathbf{x}\right)}{\sum_{z=1}^{K} \exp\left({\theta_z}^T \cdot \mathbf{x}\right)}$$

- Predicted Class:

$$k^* = \arg\max_k \text{softmax}({\theta_k}^T \cdot \mathbf{x}) = \arg\max_k {\theta_k}^T \cdot \mathbf{x}$$



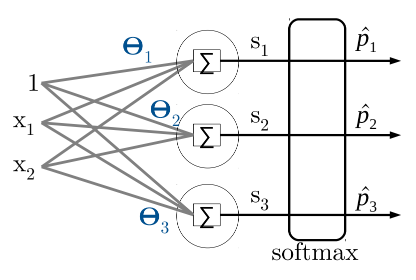
Multinomial Logistic Regression (2)

We just need to compute $K-1$ parameter vectors:

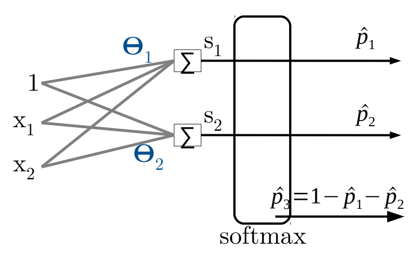
$$\hat{p}_k^{(i)} = \frac{\exp(-\theta_K)}{\exp(-\theta_K)} \cdot \frac{\exp\left(\theta_k{}^T \cdot \mathbf{x}\right)}{\sum_{z=1}^{K} \exp\left(\theta_z{}^T \cdot \mathbf{x}\right)}$$

$$= \frac{\exp\left(\left(\theta_k - \theta_K\right)^T \cdot \mathbf{x}\right)}{\sum_{z=1}^{K} \exp\left(\underbrace{\left(\theta_z - \theta_K\right)^T \cdot \mathbf{x}}_{\theta'_z}\right)}$$

$$= \frac{\exp\left(\theta_k'^{T} \cdot \mathbf{x}\right)}{1 + \sum_{z=1}^{K-1} \exp\left(\theta_z'^{T} \cdot \mathbf{x}\right)}$$



$\Downarrow$



Cross-entropy

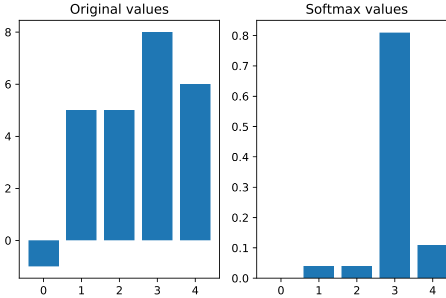
- The loss function for each sample is the cross-entropy

$$J(\theta, \mathbf{x}^{(i)}, y^{(i)}) = -\sum_k \ln \hat{p}^{(i)}_{y^{(i)}}$$

where $y^{(i)}_k = 1 \iff \mathbf{x}^{(i)} \in \text{class } k$

- We want $\hat{p}^{(i)}_{y^{(i)}}$ to be as high as possible.

- Softmax "amplifies" the most probable class.



Assignement

Show that the Multinomial Logistic Regression with $K = 2$ is equivalent to Binary Logistic Regression.

In other words, show that

$$\mathbb{P}[\mathbf{x} \in \text{class } 1] = \text{softmax}(\theta^T \cdot \mathbf{x})$$

is equivalent to the binomial case

$$\mathbb{P}[\mathbf{x} \in \text{class } 1] = \sigma(\theta^T \cdot \mathbf{x})$$

Then show that the loss function is also equivalent.

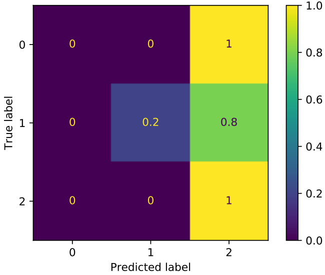
Let's code ...

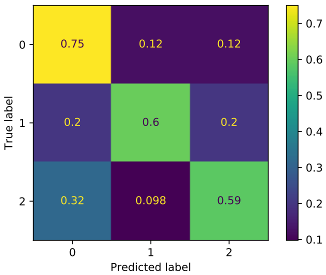
Go to notebook 03.regression_contd-and-classification.ipynb

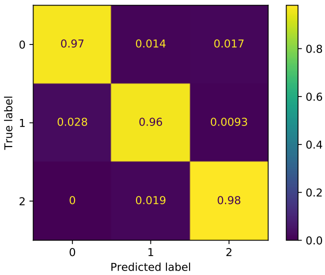
Section 6

Class imbalance and performance metrics

Confusion Matrix







Class imbalance

When there are classes with many samples and other with less samples.

How to cope with it:

- Synthetic Minority Over-Sampling TEchnique (SMOTE) [CBHK02]
  - 10 K citations!
- Others (you can explore yourself, if you want)
  - Under-sampling majority class
  - Use different weights in the loss function
  - Others: see this blog.

Classification Report

	precision	recall	f1-score	support
0	0.29	0.75	0.41	8
1	0.55	0.60	0.57	10
2	0.89	0.59	0.71	41
accuracy			0.61	59
macro avg	0.57	0.65	0.56	59
weighted avg	0.75	0.61	0.64	59

- **Precision**: 29% of samples classified as 0 are actually 0
- **Recall**: 75% of class 0 samples are correctly classified
- **Accuracy**: 61% of classifications are correct
- **Support**: 8 samples in the test set are of class 0
- **f1-score**: A combination of precision and recall:

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

$$\uparrow \text{precision}, \uparrow \text{recall} \Longrightarrow \uparrow F_1$$

Regression (continued)

- Polynomial Regression
- Variance vs. Bias Trade-Off
- Regularization
- Scaling
- Feature Selection

Classification

- Logistic Regression
- Classification Performance
- Class imbalance

To know more

- Video about feature scaling.
- More on feature selection.
- Several loss functions for classification (Video) [Mic]
- Another way of looking at Logistic Regression, based on likelihood: Sec. 4.3 of [JWHT13].

References I

[AWS] *Amazon Machine Learning Developer Guide*, https://docs.aws.amazon.com/machine-learning/.

[CBHK02] N. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, *SMOTE: Synthetic Minority Over-sampling Technique*, Journal of Artificial Intelligence Research **16** (2002), 321–357.

[Gér17] Aurélien Géron, *Hands-on machine learning with scikit-learn and tensorflow*, 2017.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, *An introduction to Statistical Learning*, vol. 7, 2013.

[KW96] Ron Kohavi and David H. Wolpert, *Bias plus variance decomposition for zero-one loss functions*, International Conference on Machine Learning (ICML96), 1996.

References II

[Mic] Microsoft, *Principles of Machine Learning | Loss Function for Classification*, https://youtu.be/r-vYJqcFxBI.

[Teo19] Jake Teo, *Data Science Documentation*, 2019.