Introduction

Jim Dowling
jdowling@kth.se

2022-11-6

Slides by Amir H. Payberah

THIS IS MY DREAM HOUSE ALRIGHT, EXCEPT IN MY DREAM IT WAS ABOUT HALF THIS PRICE.

The Housing Price Example (1/3)

- Given the dataset of m houses.

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮

- Predict the prices of other houses, as a function of the size of living area and number of bedrooms?

The Housing Price Example (2/3)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮

$$\mathbf{x}^{(1)} = \begin{bmatrix} 2104 \\ 3 \end{bmatrix} \quad y^{(1)} = 400 \qquad \mathbf{x}^{(2)} = \begin{bmatrix} 1600 \\ 3 \end{bmatrix} \quad y^{(2)} = 330 \qquad \mathbf{x}^{(3)} = \begin{bmatrix} 2400 \\ 3 \end{bmatrix} \quad y^{(3)} = 369$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)\intercal} \\ \mathbf{x}^{(2)\intercal} \\ \mathbf{x}^{(3)\intercal} \\ \vdots \end{bmatrix} = \begin{bmatrix} 2104 & 3 \\ 1600 & 3 \\ 2400 & 3 \\ \vdots & \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ \vdots \end{bmatrix}$$

- $\mathbf{x}^{(i)} \in \mathbb{R}^2$: $x_1^{(i)}$ is the living area, and $x_2^{(i)}$ is the number of bedrooms of the ith house in the training set.

The Housing Price Example (3/3)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮

- Predict the prices of other houses $\hat{y}$ as a function of the size of their living areas x_1, and number of bedrooms x_2, i.e., $\hat{y} = f(x_1, x_2)$

- E.g., what is $\hat{y}$, if $x_1 = 4000$ and $x_2 = 4$?

- As an initial choice: $\hat{y} = f_w(\mathbf{x}) = w_1 x_1 + w_2 x_2$

Linear Regression

Linear Regression (1/2)

- Our goal: to build a system that takes input $\mathbf{x} \in \mathbb{R}^{\mathtt{n}}$ and predicts output $\hat{\mathtt{y}} \in \mathbb{R}$.
- In linear regression, the output $\hat{\mathtt{y}}$ is a linear function of the input $\mathbf{x}$.

$$\hat{\mathtt{y}} = \mathtt{f}_{\mathtt{w}}(\mathbf{x}) = \mathtt{w}_1\mathtt{x}_1 + \mathtt{w}_2\mathtt{x}_2 + \cdots + \mathtt{w}_{\mathtt{n}}\mathtt{x}_{\mathtt{n}}$$

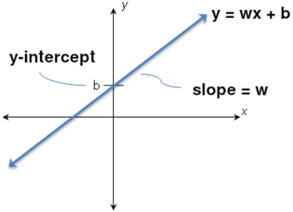
$$\hat{\mathtt{y}} = \mathbf{w}^{\intercal}\mathbf{x}$$

  - $\hat{\mathtt{y}}$: the predicted value
  - $\mathtt{n}$: the number of features
  - $\mathtt{x}_{\mathtt{i}}$: the $\mathtt{i}$th feature value
  - $\mathtt{w}_{\mathtt{j}}$: the $\mathtt{j}$th model parameter ($\mathbf{w} \in \mathbb{R}^{\mathtt{n}}$)

Linear Regression (2/2)

- Linear regression often has one additional parameter, called intercept b:

$$\hat{y} = \mathbf{w}^\intercal \mathbf{x} + b$$



- Instead of adding the bias parameter b, we can augment $\mathbf{x}$ with an extra entry that is always set to 1.

$$\hat{y} = f_w(\mathbf{x}) = w_0x_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n, \text{ where } x_0 = 1$$

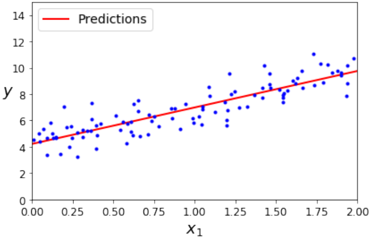
Linear Regression - Model Parameters

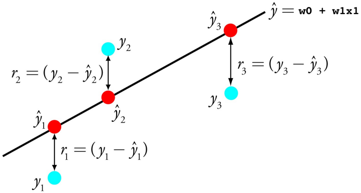
$$\hat{y} = f_{w}(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

- Parameters $\mathbf{w} \in \mathbb{R}^n$ are values that control the behavior of the model.
- $\mathbf{w}$ are a set of weights that determine how each feature affects the prediction.
  - $w_i > 0$: increasing the value of the feature x_i, increases the value of our prediction $\hat{y}$.
  - $w_i < 0$: increasing the value of the feature x_i, decreases the value of our prediction $\hat{y}$.
  - $w_i = 0$: the value of the feature x_i, has no effect on the prediction $\hat{y}$.

How can you learn Model Parameters $\mathbf{w}$?

Linear Regression - Cost Function (1/2)

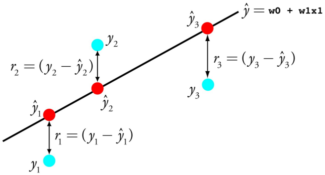



$\hat{y}$ = w0 + w1x1
$\hat{y}_3$
y_2
$r_2 = (y_2 - \hat{y}_2)$
$r_3 = (y_3 - \hat{y}_3)$
y_3
$\hat{y}_2$
$\hat{y}_1$
$r_1 = (y_1 - \hat{y}_1)$
y_1

- One reasonable model should make $\hat{\mathtt{y}}$ close to $\mathtt{y}$, at least for the training dataset.
- Residual: the difference between the dependent variable $\mathtt{y}$ and the predicted value $\hat{\mathtt{y}}$.

$$\mathtt{r}^{(\mathtt{i})} = \mathtt{y}^{(\mathtt{i})} - \hat{\mathtt{y}}^{(\mathtt{i})}$$

Linear Regression - Cost Function (2/2)



$\hat{y} = $ w0 + w1x1
$\hat{y}_3$
y_2
$r_2 = (y_2 - \hat{y}_2)$
$r_3 = (y_3 - \hat{y}_3)$
y_3
$\hat{y}_2$
$\hat{y}_1$
$r_1 = (y_1 - \hat{y}_1)$
y_1

- Cost function $\mathtt{J}(\mathbf{w})$
    - For each value of the $\mathbf{w}$, it measures how close the $\hat{\mathtt{y}}^{(\mathtt{i})}$ is to the corresponding $\mathtt{y}^{(\mathtt{i})}$.
    - We can define $\mathtt{J}(\mathbf{w})$ as the mean squared error (MSE):

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) = \frac{1}{\mathtt{m}} \sum_{\mathtt{i}}^{\mathtt{m}} (\hat{\mathtt{y}}^{(\mathtt{i})} - \mathtt{y}^{(\mathtt{i})})^2$$

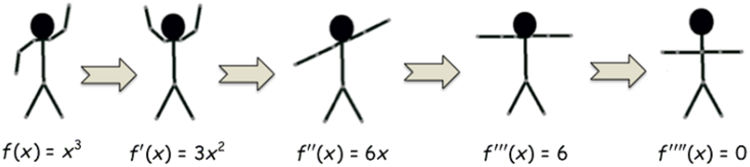
$$= \mathtt{E}[(\hat{\mathtt{y}} - \mathtt{y})^2] = \frac{1}{\mathtt{m}} ||\hat{\mathbf{y}} - \mathbf{y}||_2^2$$

How can you learn Model Parameters?

- We want to choose $\mathbf{w}$ so as to minimize $\mathtt{J}(\mathbf{w})$.
- Two approaches to find $\mathbf{w}$:
  - Normal equation - closed form solution
  - Gradient descent - iterative optimization

Normal Equation

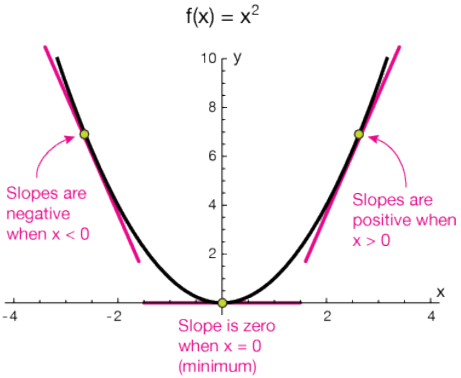
Derivatives and Gradient (1/4)



[https://mathequality.wordpress.com/2012/09/26/derivative-dance-gangnam-style/]

Derivatives and Gradient (2/4)

- The first derivative of $\mathtt{f}(\mathtt{x})$, shown as $\mathtt{f}'(\mathtt{x})$, shows the slope of the tangent line to the function at the poa $\mathtt{x}$.
- $\mathtt{f}(\mathtt{x}) = \mathtt{x}^2 \Rightarrow \mathtt{f}'(\mathtt{x}) = 2\mathtt{x}$
- If $\mathtt{f}(\mathtt{x})$ is increasing, then $\mathtt{f}'(\mathtt{x}) > 0$
- If $\mathtt{f}(\mathtt{x})$ is decreasing, then $\mathtt{f}'(\mathtt{x}) < 0$
- If $\mathtt{f}(\mathtt{x})$ is at local minimum/maximum, then $\mathtt{f}'(\mathtt{x}) = 0$



Derivatives and Gradient (3/4)

- What if a function has multiple arguments, e.g., $f(x_1, x_2, \cdots, x_n)$

- Partial derivatives: the derivative with respect to a particular argument.
  - $\frac{\partial f}{\partial x_1}$, the derivative with respect to x_1
  - $\frac{\partial f}{\partial x_2}$, the derivative with respect to x_2

- $\frac{\partial f}{\partial x_i}$: shows how much the function f will change, if we change x_i.

- Gradient: the vector of all partial derivatives for a function f.

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Derivatives and Gradient (4/4)

- What is the gradient of $f(x_1, x_2, x_3) = x_1 - x_1 x_2 + x_3^2$?

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}(x_1 - x_1 x_2 + x_3^2) \\ \frac{\partial}{\partial x_2}(x_1 - x_1 x_2 + x_3^2) \\ \frac{\partial}{\partial x_3}(x_1 - x_1 x_2 + x_3^2) \end{bmatrix} = \begin{bmatrix} 1 - x_2 \\ -x_1 \\ 2x_3 \end{bmatrix}$$

Normal Equation

- To minimize $\mathtt{J}(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}}\mathtt{J}(\mathbf{w}) = 0$

$$\hat{y} = \mathbf{w}^{\intercal}\mathbf{x}$$

$$\dots$$

$$\dots$$

$$\Rightarrow \mathbf{w} = (\mathbf{X}^{\intercal}\mathbf{X})^{-1}\mathbf{X}^{\intercal}\mathbf{y}$$

Normal Equation - Example (1/7)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540

- Predict the value of $\hat{y}$, when $x_1 = 4000$ and $x_2 = 4$.
- We should find w_0, w_1, and w_2 in $\hat{y} = w_0 + w_1 x_1 + w_2 x_2$.
- $\mathbf{w} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y}$.

Normal Equation - Example (2/7)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540

$$\mathbf{X} = \left[\begin{array}{c|cc} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{array}\right] \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix}$$

Normal Equation - Example (3/7)

$$\mathbf{X}^\intercal\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2104 & 1600 & 2400 & 1416 & 3000 \\ 3 & 3 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{bmatrix} = \begin{bmatrix} 5 & 10520 & 15 \\ 10520 & 23751872 & 33144 \\ 15 & 33144 & 47 \end{bmatrix}$$

Normal Equation - Example (4/7)

$$(\mathbf{X}^\intercal\mathbf{X})^{-1} = \begin{bmatrix} 4.90366455\text{e}+00 & 7.48766737\text{e}-04 & -2.09302326\text{e}+00 \\ 7.48766737\text{e}-04 & 2.75281889\text{e}-06 & -2.18023256\text{e}-03 \\ -2.09302326\text{e}+00 & -2.18023256\text{e}-03 & 2.22674419\text{e}+00 \end{bmatrix}$$

Normal Equation - Example (5/7)

$$\mathbf{X}^\mathsf{T}\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2104 & 1600 & 2400 & 1416 & 3000 \\ 3 & 3 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix} = \begin{bmatrix} 1871 \\ 4203712 \\ 5921 \end{bmatrix}$$

Normal Equation - Example (6/7)

$$\mathbf{w} = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathrm{y} = \begin{bmatrix} 4.90366455\mathrm{e}+00 & 7.48766737\mathrm{e}-04 & -2.09302326\mathrm{e}+00 \\ 7.48766737\mathrm{e}-04 & 2.75281889\mathrm{e}-06 & -2.18023256\mathrm{e}-03 \\ -2.09302326\mathrm{e}+00 & -2.18023256\mathrm{e}-03 & 2.22674419\mathrm{e}+00 \end{bmatrix} \begin{bmatrix} 1871 \\ 4203712 \\ 5921 \end{bmatrix}$$

$$= \begin{bmatrix} -7.04346018e+01 \\ 6.38433756e-02 \\ 1.03436047e+02 \end{bmatrix}$$

Normal Equation - Example (7/7)

- Predict the value of $\mathtt{y}$, when $\mathtt{x}_1 = 4000$ and $\mathtt{x}_2 = 4$.

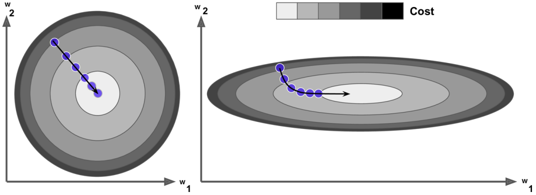
$$\hat{\mathtt{y}} = -7.04346018\mathrm{e}+01 + 6.38433756\mathrm{e}-02 \times 4000 + 1.03436047\mathrm{e}+02 \times 4 \approx 599$$

Gradient Descent

Gradient Descent (1/2)

- Gradient descent is a generic optimization algorithm capable of finding optimal solutions to a wide range of problems.

- The idea: to tweak parameters iteratively in order to minimize a cost function.

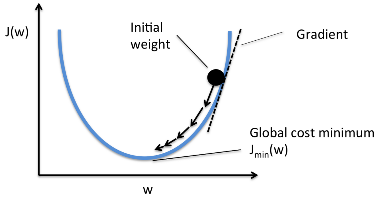


Gradient Descent (2/2)

- Suppose you are lost in the mountains in a dense fog.
- You can only feel the slope of the ground below your feet.
- A strategy to get to the bottom of the valley is to go downhill in the direction of the steepest slope.

Gradient Descent - Iterative Optimization Algorithm

- Choose a starting point, e.g., filling **w** with random values.
- If the stopping criterion is true return the current solution, otherwise continue.
- Find a descent direction, a direction in which the function value decreases near the current point.
- Determine the step size, the length of a step in the given direction.



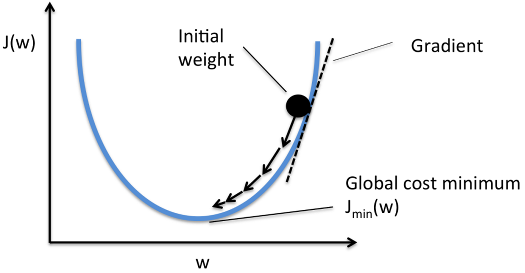
Gradient Descent - Key Points

- Stopping criterion
- Descent direction
- Step size (learning rate)

Gradient Descent - Stopping Criterion

- The cost function minimum property: the gradient has to be zero.

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$$

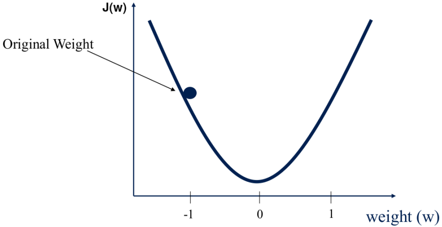


Gradient Descent - Descent Direction (1/2)

- Direction in which the function value decreases near the current point.
- Find the direction of descent (slope).
- Example:

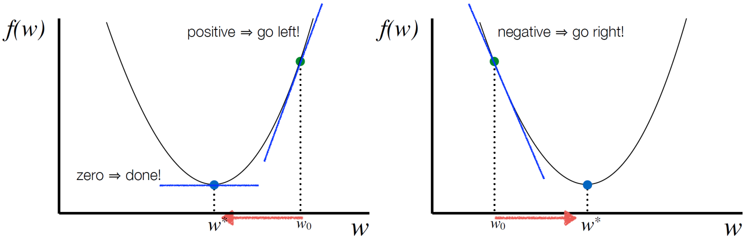
$$J(w) = w^2$$

$$\frac{\partial J(w)}{\partial w} = 2w = -2 \text{ at } w = -1$$



Gradient Descent - Descent Direction (2/2)

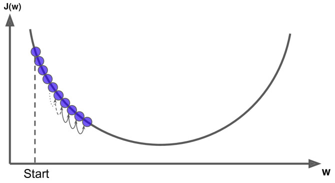
- Follow the opposite direction of the slope.



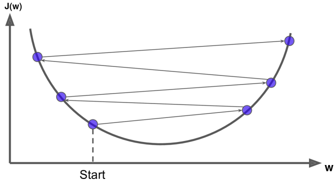
Gradient Descent - Learning Rate

- Learning rate: the length of steps.

- If it is too small: many iterations to converge.



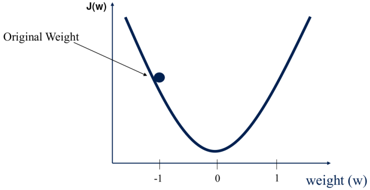
- If it is too high: the algorithm might diverge.



Gradient Descent - How to Learn Model Parameters $\mathbf{w}$?

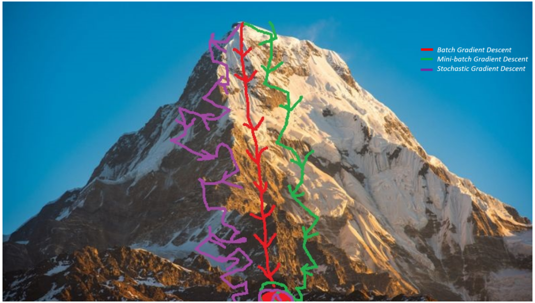
- Goal: find $\mathbf{w}$ that minimizes $\mathtt{J}(\mathbf{w}) = \sum_{\mathtt{i}=1}^{\mathtt{m}}(\mathbf{w}^\intercal \mathbf{x}^{(\mathtt{i})} - \mathtt{y}^{(\mathtt{i})})^2$.

- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial \mathtt{J}(\mathbf{w})}{\partial \mathtt{w}}$
  2. Choose a step size η
  3. Update the parameters: $\mathtt{w}^{(\mathtt{next})} = \mathtt{w} - \eta \frac{\partial \mathtt{J}(\mathbf{w})}{\partial \mathtt{w}}$ (should be done for all parameters simultanously)



Gradient Descent - Different Algorithms

- Batch gradient descent (all samples)
- Stochastic gradient descent (1 sample)
- Mini-batch gradient descent (a mini-batch of samples - e.g., 200 samples)



[https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3]

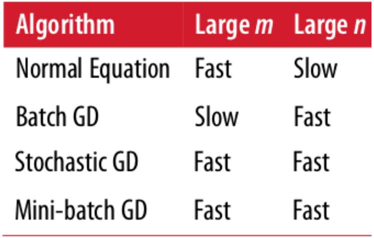
Mini-Batch Gradient Descent

- Batch gradient descent: at each step, it computes the gradients based on the full training set.

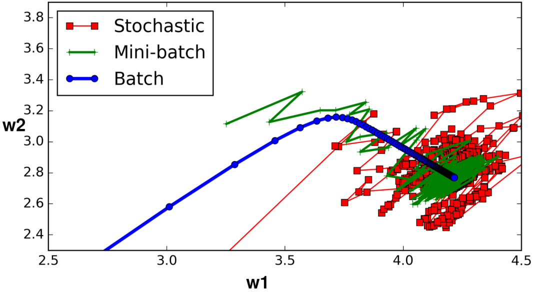
- Stochastic gradient descent: at each step, it computes the gradients based on just one instance.

- Mini-batch gradient descent: at each step, it computes the gradients based on small random sets of instances called mini-batches.

Comparison of Algorithms for Linear Regression



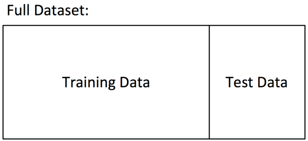
Algorithm	Large *m*	Large *n*
Normal Equation	Fast	Slow
Batch GD	Slow	Fast
Stochastic GD	Fast	Fast
Mini-batch GD	Fast	Fast



Generalization

Training Data and Test Data

- Split data into a training set and a test set.
- Use training set when training a machine learning model.
  - Compute training error on the training set.
  - Try to reduce this training error.


Full Dataset:

- Use test set to measure the accuracy of the model.
  - Test error is the error when you run the trained model on test data (new data).

Generalization

- Generalization: make a model that performs well on test data.
    - Have a small test error.

- Challenges
    1. Make the training error small.
    2. Make the gap between training and test error small.

More About The Test Error

- The test error is defined as the expected value of the error on test set.

$$\begin{aligned} \mathtt{MSE} &= \frac{1}{\mathtt{k}} \sum_{\mathtt{i}}^{\mathtt{k}} (\hat{\mathtt{y}}^{(\mathtt{i})} - \mathtt{y}^{(\mathtt{i})})^2, \ \mathtt{k}\text{: the num. of instances in the test set} \\ &= \mathtt{E}[(\hat{\mathtt{y}} - \mathtt{y})^2] \end{aligned}$$

- A model's test error can be expressed as the sum of bias and variance.

$$\mathtt{E}[(\hat{\mathtt{y}} - \mathtt{y})^2] = \mathtt{Bias}[\hat{\mathtt{y}}, \mathtt{y}]^2 + \mathtt{Var}[\hat{\mathtt{y}}] + \varepsilon^2$$

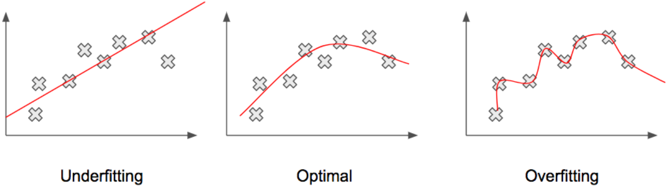
"BIAS" GUESSES WRONG IN TRAINING
"VARIANCE" GUESSES WRONG IN TESTING
ITS GOING TO BE OK
imgflip.com
@scott.ai

Bias and Underfitting

- Bias: the expected deviation from the true value of the function.

$$\texttt{Bias}[\hat{y}, y] = \texttt{E}[\hat{y}] - y$$

- A high-bias model is most likely to underfit the training data.
  - High error value on the training set.
- Underfitting happens when the model is too simple to learn the underlying structure of the data.



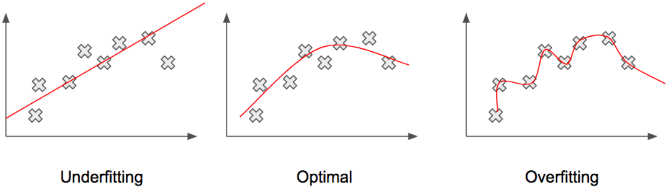
Variance and Overfitting

- Variance: how much a model changes if you train it on a different training set.

$$\mathtt{Var}[\hat{y}] = \mathtt{E}[(\hat{y} - \mathtt{E}[\hat{y}])^2]$$

- A high-variance model is most likely to overfit the training data.
  - The gap between the training error and test error is too large.

- Overfitting happens when the model is too complex relative to the amount and noisiness of the training data.

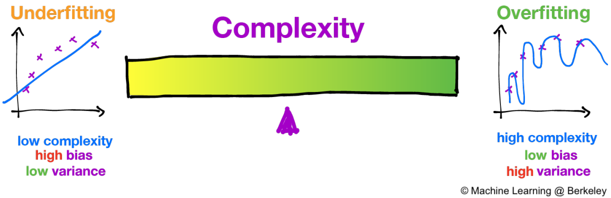


The Bias/Variance Tradeoff (1/2)

- Assume a model with two parameters $\mathtt{w}_0$ (intercept) and $\mathtt{w}_1$ (slope): $\hat{\mathtt{y}} = \mathtt{w}_0 + \mathtt{w}_1\mathtt{x}$
- They give the learning algorithm two degrees of freedom.
- We tweak both the $\mathtt{w}_0$ and $\mathtt{w}_1$ to adapt the model to the training data.
- If we forced $\mathtt{w}_0 = 0$, the algorithm would have only one degree of freedom and would have a much harder time fitting the data properly.

The Bias/Variance Tradeoff (2/2)

- Increasing degrees of freedom will typically increase its variance and reduce its bias.
- Decreasing degrees of freedom increases its bias and reduces its variance.
- This is why it is called a tradeoff.



[https://ml.berkeley.edu/blog/2017/07/13/tutorial-4]

Regularization (1/2)

- One way to reduce the risk of overfitting is to have fewer degrees of freedom.
- Regularization is a technique to reduce the risk of overfitting.
- For a linear model, regularization is achieved by constraining the weights of the model.

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \lambda \mathtt{R}(\mathbf{w})$$

Regularization (2/2)

- Lasso regression (*l*1): $\mathtt{R}(\mathbf{w}) = \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} |\mathtt{w}_\mathtt{i}|$ is added to the cost function:

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} |\mathtt{w}_\mathtt{i}|$$

- Ridge regression (*l*2): $\mathtt{R}(\mathbf{w}) = \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} \mathtt{w}_\mathtt{i}^2$ is added to the cost function.

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} \mathtt{w}_\mathtt{i}^2$$

- ElasticNet: a middle ground between *l*1 and *l*2 regularization.

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \alpha\lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} |\mathtt{w}_\mathtt{i}| + (1 - \alpha)\lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} \mathtt{w}_\mathtt{i}^2$$

Hyperparameters

Hyperparameters and Validation Sets (1/2)

- Hyperparameters are settings that we can use to control the behavior of a learning algorithm.

- The values of hyperparameters are not adapted by the learning algorithm itself.
  - E.g., the α and λ values for regularization.

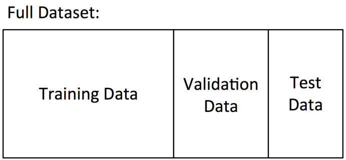
- We do not learn the hyperparameter.
  - It is not appropriate to learn that hyperparameter on the training set.
  - If learned on the training set, such hyperparameters would always result in overfitting.

Hyperparameters and Validation Sets (2/2)

- To find hyperparameters, we need a validation set of examples that the training algorithm does not observe.

- We construct the validation set from the training data (not the test data).

- We split the training data into two disjoint subsets:
  1. One is used to learn the parameters.
  2. The other one (the validation set) is used to estimate the test error during or after training, allowing for the hyperparameters to be updated accordingly.



Full Dataset:

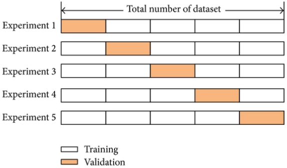
Training Data

Validation Data

Test Data

Cross-Validation

- Cross-validation: a technique to avoid wasting too much training data in validation sets.
- The training set is split into complementary subsets.
- Each model is trained against a different combination of these subsets and validated against the remaining parts.
- Once the model type and hyperparameters have been selected, a final model is trained using these hyperparameters on the full training set, and the test error is measured on the test set.



Regression Summary

- Linear regression model $\hat{y} = \mathbf{w}^\intercal \mathbf{x}$
  - Learning parameters $\mathbf{w}$
  - Cost function $\mathtt{J}(\mathbf{w})$
  - Learn parameters: normal equation, gradient descent (batch, stochastic, mini-batch)
- Generalization
  - Overfitting vs. underfitting
  - Bias vs. variance
  - Regularization: Lasso regression, Ridge regression, ElasticNet
- Hyperparameters and cross-validation

Classification

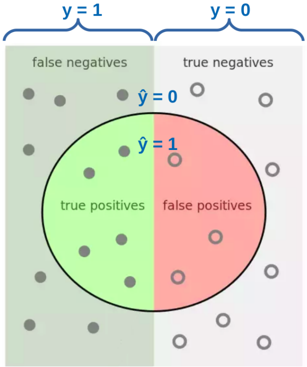
Evaluation of Classification Models (1/3)

- In a classification problem, there exists a true output y and a model-generated predicted output $\hat{y}$ for each data point.

- The results for each instance point can be assigned to one of four categories:
  - True Positive (TP)
  - True Negative (TN)
  - False Positive (FP)
  - False Negative (FN)

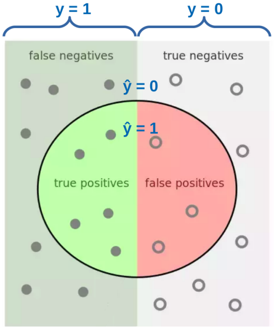
Evaluation of Classification Models (2/3)

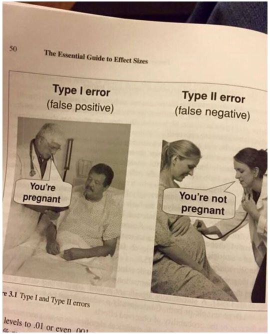
- True Positive (TP): the label y is positive and prediction $\hat{y}$ is also positive.
- True Negative (TN): the label y is negative and prediction $\hat{y}$ is also negative.



Evaluation of Classification Models (3/3)

- False Positive (FP): the label y is negative but prediction $\hat{y}$ is positive (type I error).
- False Negative (FN): the label y is positive but prediction $\hat{y}$ is negative (type II error).




50
The Essential Guide to Effect Sizes
Type I error
(false positive)
Type II error
(false negative)
You're pregnant
You're not pregnant
re 3.1 Type I and Type II errors
levels to .01 or even .001

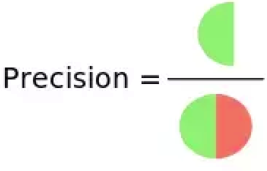
Why Pure Accuracy Is Not A Good Metric?

- Accuracy: how close the prediction is to the true value.
- Assume a highly unbalanced dataset
- E.g., a dataset where 95% of the data points are not fraud and 5% of the data points are fraud.
- A a naive classifier that predicts not fraud, regardless of input, will be 95% accurate.
- For this reason, metrics like precision and recall are typically used.

Precision

- It is the accuracy of the positive predictions.

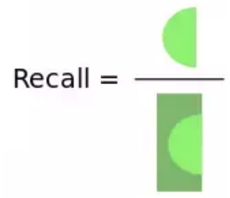
$$\mathtt{Precision} = \mathtt{p}(\mathtt{y} = 1 \mid \hat{\mathtt{y}} = 1) = \frac{\mathtt{TP}}{\mathtt{TP} + \mathtt{FP}}$$



Recall

- Is is the ratio of positive instances that are correctly detected by the classifier.
- Also called sensitivity or true positive rate (TPR).

$$\mathtt{Recall} = \mathrm{p}(\hat{\mathrm{y}} = 1 \mid \mathrm{y} = 1) = \frac{\mathtt{TP}}{\mathtt{TP} + \mathtt{FN}}$$



F1 Score

- $F1$ score: combine precision and recall into a single metric.

- The $F1$ score is the harmonic mean of precision and recall.

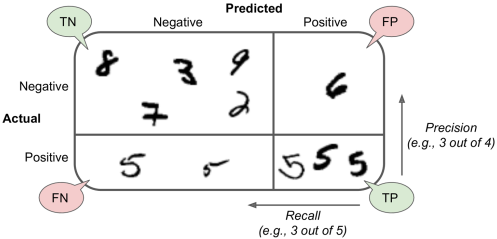
- Whereas the regular mean treats all values equally, the harmonic mean gives much more weight to low values.

- $F1$ only gets high score if both recall and precision are high.

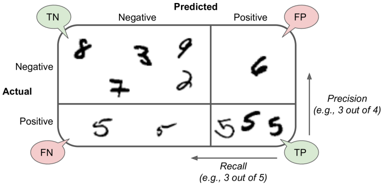
$$\texttt{F1} = \frac{2}{\frac{1}{\texttt{precision}} + \frac{1}{\texttt{recall}}}$$

Confusion Matrix

- The confusion matrix is $K \times K$, where K is the number of classes.
- It shows the number of correct and incorrect predictions made by the classification model compared to the actual outcomes in the data.



Confusion Matrix - Example



$$\mathrm{TP} = 3, \mathrm{TN} = 5, \mathrm{FP} = 1, \mathrm{FN} = 2$$

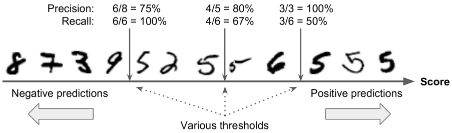
$$\mathtt{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{3}{3+1} = \frac{3}{4}$$

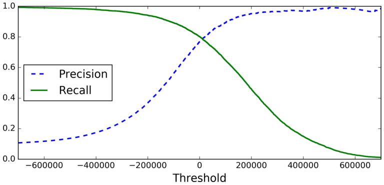
$$\mathtt{Recall}\ (\mathrm{TPR}) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{3}{3+2} = \frac{3}{5}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} = \frac{1}{5+1} = \frac{5}{6}$$

Precision-Recall Tradeoff

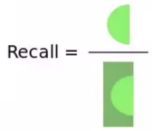
- Precision-recall tradeoff: increasing precision reduces recall, and vice versa.
- Assume a classifier that detects number 5 from the other digits.
  - If an instance score is greater than a threshold, it assigns it to the positive class, otherwise to the negative class.
- Raising the threshold (move it to the arrow on the right), the false positive (the 6) becomes a true negative, thereby increasing precision.
- Lowering the threshold increases recall and reduces precision.

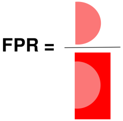




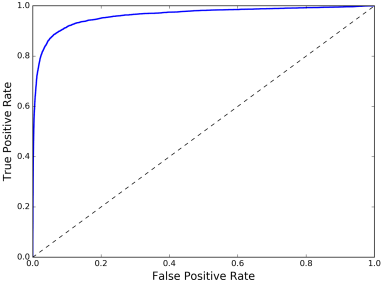
The ROC Curve (1/2)

- True positive rate (TPR) (recall): $p(\hat{y} = 1 \mid y = 1)$
- False positive rate (FPR): $p(\hat{y} = 1 \mid y = 0)$



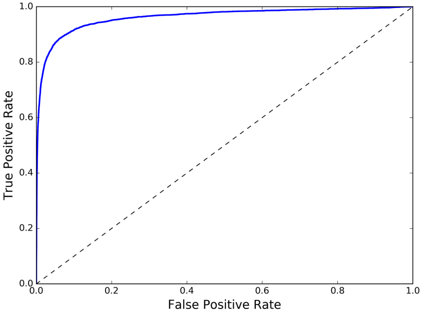


- The receiver operating characteristic (ROC) curves summarize the trade-off between the TPR and FPR for a model using different probability thresholds.



The ROC Curve (2/2)

- Here is a tradeoff: the higher the TPR, the more FPR the classifier produces.
- The dotted line represents the ROC curve of a purely random classifier.
- A good classifier moves toward the top-left corner.
- Area under the curve (AUC)



Decision Trees

Buying Computer Example (1/3)

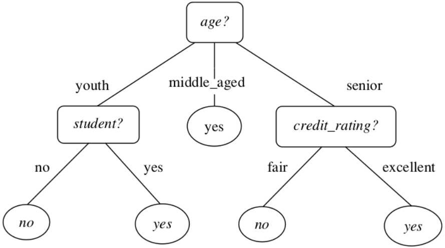
- Given the dataset of $\mathtt{m}$ people.

id	age	income	student	credit rating	buys computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middleage	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
⋮	⋮	⋮	⋮	⋮	⋮

- Predict if a new person buys a computer?
- Given an instance $\mathbf{x}^{(i)}$, e.g., $x_1^{(i)} = \mathtt{senior}$, $x_2^{(i)} = \mathtt{medium}$, $x_3^{(i)} = \mathtt{no}$, and $x_4^{(i)} = \mathtt{fair}$, then $y^{(i)} = ?$

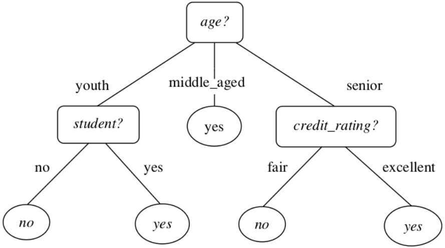
Buying Computer Example (2/3)

id	age	income	student	credit rating	buys computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middleage	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
⋮	⋮	⋮	⋮	⋮	⋮



Buying Computer Example (3/3)

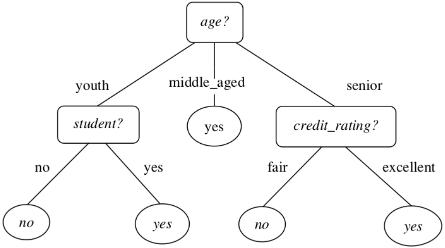
- Given an input instance $\mathbf{x}^{(i)}$, for which the class label $y^{(i)}$ is unknown.
- The attribute values of the input (e.g., age or income) are tested.
- A path is traced from the root to a leaf node, which holds the class prediction for that input.
- E.g., input $\mathbf{x}^{(i)}$ with $x_1^{(i)} = \texttt{senior}$, $x_2^{(i)} = \texttt{medium}$, $x_3^{(i)} = \texttt{no}$, and $x_4^{(i)} = \texttt{fair}$.



Decision Trees

Decision Tree

- A decision tree is a flowchart-like tree structure.
  - The topmost node: represents the root
  - Each internal node: denotes a test on an attribute
  - Each branch: represents an outcome of the test
  - Each leaf: holds a class label



Training Algorithm (1/2)

- Decision trees are constructed in a top-down recursive divide-and-conquer manner.
- The algorithm is called with the following parameters.
  - Data partition $\mathtt{D}$: initially the complete set of training data and labels $\mathtt{D} = (\mathbf{X}, \mathbf{y})$.
  - Feature list: list of features $\{\mathbf{x}_1^{(i)}, \cdots, \mathbf{x}_n^{(i)}\}$ of each data instance $\mathbf{x}^{(i)}$.
  - Feature selection method: determines the splitting criterion.

Training Algorithm (2/2)

- 1. The tree starts as a single node, $\mathtt{N}$, representing the training data instances $\mathtt{D}$.
- 2. If all instances $\mathbf{x}$ in $\mathtt{D}$ are all of the same class, then node $\mathtt{N}$ becomes a leaf.
- 3. The algorithm calls feature selection method to determine the splitting criterion.
  - Indicates (i) the splitting feature $\mathtt{x}_\mathtt{k}$, and (ii) a split-point or a splitting subset.
  - The instances in $\mathtt{D}$ are partitioned accordingly.
- 4. The algorithm repeats the same process recursively to form a decision tree.

Training Algorithm - Termination Conditions

- The training algorithm stops only when any one of the following conditions is true.
- 1. All the instances in partition D at a node N belong to the same class.
  - It is labeled with that class.
- 2. No remaining features on which the instances may be further partitioned.
- 3. There are no instances for a given branch, that is, a partition D_j is empty.
- In conditions 2 and 3:
  - Convert node N into a leaf.
  - Label it either with the most common class in D.
  - Or, the class distribution of the node tuples may be stored.

Ensembles

Wisdom of the Crowd

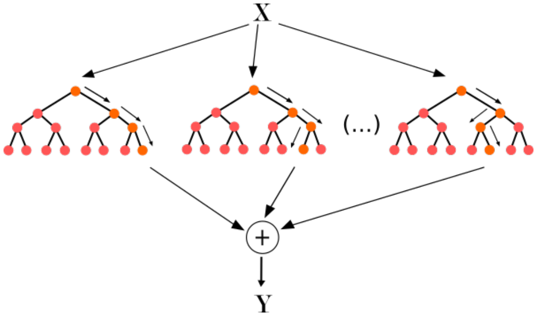
- Ask a complex question to thousands of random people, then aggregate their answers.
- In many cases, this aggregated answer is better than an expert's answer.
- This is called the wisdom of the crowd.
- Similarly, the aggregated estimations of a group of estimators (e.g., classifiers or regressors), often gets better estimations than with the best individual estimator.
- A group of estimators is an ensemble, and this technique is called Ensemble Learning.

Ensemble Learning

- Two main categories of ensemble learning algorithms.
- Bagging
  - Use the same training algorithm for every estimator, but to train them on different random subsets of the training set.
  - E.g., random forest
- Boosting
  - Train estimators sequentially, each trying to correct its predecessor.
  - E.g., adaboost and gradient boosting

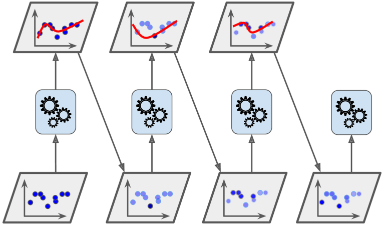
Random Forest

- Random forest builds multiple decision trees that are most of the time trained with the bagging method.
- It, then, merges the trees together to get a more accurate and stable prediction.



AdaBoost

- AdaBoost: train a new estimator by paying more attention to the training instances that the predecessor underfitted.
- Each estimator is trained on a random subset of the total training set.
- AdaBoost assigns a weight to each training instance, which determines the probability that each instance should appear in the training set.



Gradient Boosting (1/3)

- Just like AdaBoost, Gradient Boosting works by sequentially adding estimators to an ensemble, each one correcting its predecessor.
- However, instead of tweaking the instance weights at every iteration, this method tries to fit the new estimator to the residual errors made by the previous estimator.

Gradient Boosting (2/3)

- Let's go through a regression example using Gradient Boosted Regression Trees.

- Fit the first estimator on the training set.

```
tree_reg1 = DecisionTreeRegressor(max_depth=2)
tree_reg1.fit(X, y)
```

- Now train the second estimator on the residual errors made by the first estimator.

```
y2 = y - tree_reg1.predict(X)
tree_reg2 = DecisionTreeRegressor(max_depth=2)
tree_reg2.fit(X, y2)
```

Gradient Boosting (3/3)

- Then we train the third estimator on the residual errors made by the second estimator.

```
y3 = y2 - tree_reg2.predict(X)
tree_reg3 = DecisionTreeRegressor(max_depth=2)
tree_reg3.fit(X, y3)
```

- Now we have an ensemble containing three trees.
- It can make predictions on a new instance simply by adding up the predictions of all the trees.

```
y_pred = sum(tree.predict(X_new) for tree in (tree_reg1, tree_reg2, tree_reg3))
```

Summary

- Decision tree
  - Top-down training algorithm
  - Termination condition
- Ensemble models
  - Bagging: random forest
  - Boosting: AdaBoost, Gradient Boosting

Reference

- Ian Goodfellow et al., Deep Learning (Ch. 4, 5)
- Aurélien Géron, Hands-On Machine Learning (Ch. 2, 3, 4)

Questions?