Pythonで行う機械学習プログラム

Lesson_0 Python の基礎: インストール、主要ライブラリ操作（pandas, numpy, matplotlib）

Lesson_1 データセット取り扱いと機械学習基礎

Lesson_2 Python: 機械学習（分類）評価と応用

Lesson_3 Python: 機械学習（回帰）評価と応用

Lesson_4 Python: 機械学習 前処理、最適化

Lesson_5 Python: OpenCV による画像処理

Lesson_6 Python: DL 画像判別（１） MLP と CNN

Lesson_7 Python: DL 画像判別（２） 転移学習、学習データの再利用、結果の保存

機械学習プロセス


課題設定
・分析から得たいものを決定
データ取得
・ログ
・アンケート
・WebAPI
etc..
データ前処理
・欠損値処理
・特徴量選択
・次元削除
etc..
手法選択
・SVM
・決定木
・k-NN
etc..
モデル学習
・手法選択で選択をしたモデルを使用し学習させる
モデル評価
・正解率
・適合率
・再現率
・汎化性能の評価
etc..

今回取り扱うデータセット

天体の構成がパルサー星かどうかを判別する → True or False の**2値問題**

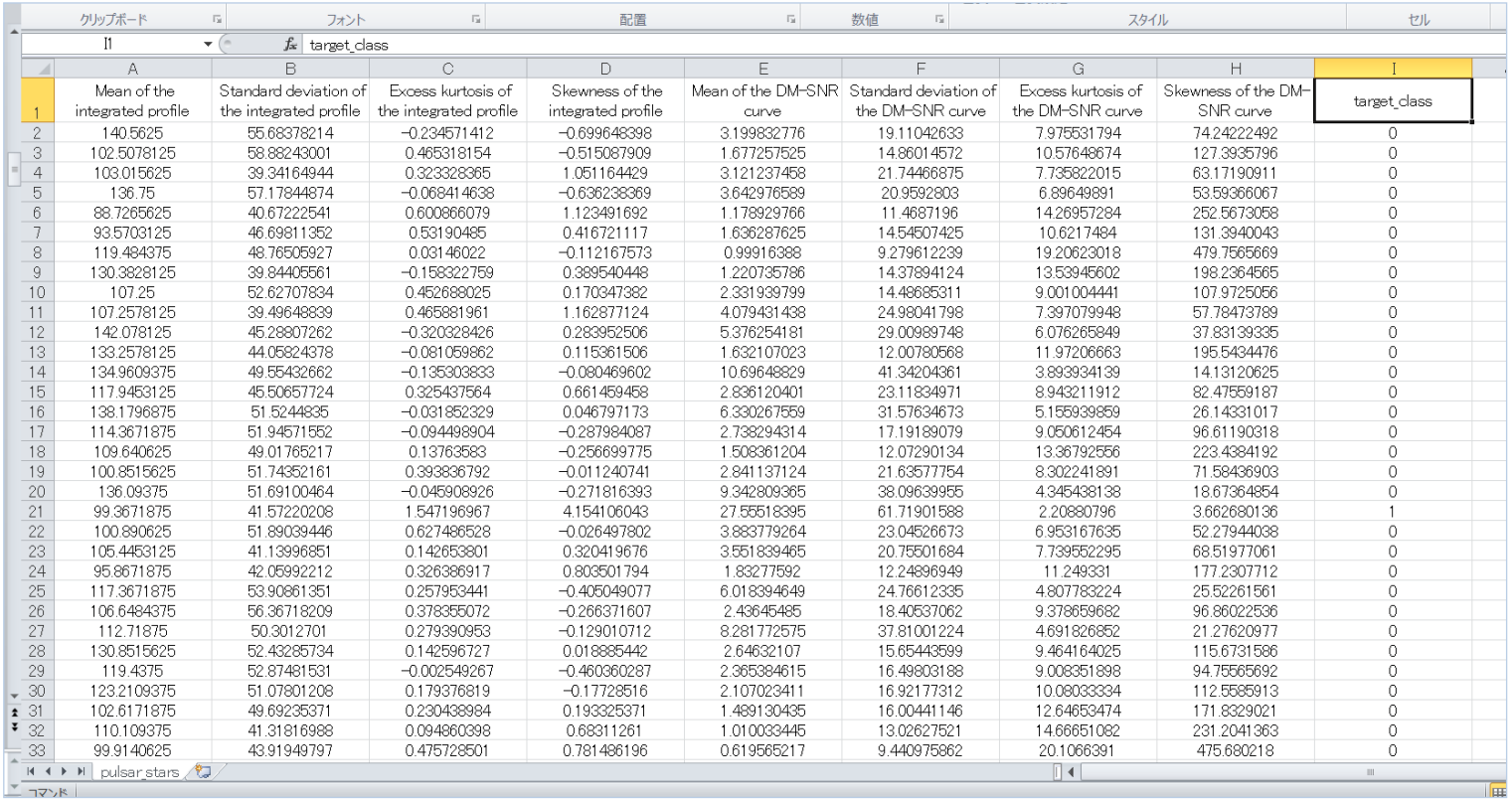
(1) Mean of the integrated profile
(2) Standard deviation of the integrated profile
(3) Excess kurtosis of the integrated profile
(4) Skewness of the integrated profile
(5) Mean of the DM-SNR curve
(6) Standard deviation of the DM-SNR curve
(7) Excess kurtosis of the DM-SNR curve
(8) Skewness of the DM-SNR curve

上記8つの説明変数を基に、パルサーかどうか（目的変数）の判別を行う。

→ 機械学習を行う上では**その分野の専門知識は一切なくても行える**。
（つまり、経験や勘に頼らなくてもよい。時には専門知識による常識が、機械学習の妨げになることもある）

機械学習デモ＋実習

今回準備したデータファイル：Pulsar_stars.csv



	A	B	C	D	E	F	G	H	I
1	Mean of the integrated profile	Standard deviation of the integrated profile	Excess kurtosis of the integrated profile	Skewness of the integrated profile	Mean of the DM-SNR curve	Standard deviation of the DM-SNR curve	Excess kurtosis of the DM-SNR curve	Skewness of the DM-SNR curve	target_class
2	140.5625	55.68378214	-0.234571412	-0.699648398	3.199832776	19.11042633	7.975531794	74.24222492	0
3	102.5078125	58.88243001	0.465318154	-0.515087909	1.677257525	14.86014572	10.57648674	127.3935796	0
4	103.015625	39.34164944	0.323328365	1.051164429	3.121237458	21.74466875	7.735822015	63.17190911	0
5	136.75	57.17844874	-0.068414638	-0.636238369	3.642976589	20.9592803	6.89649891	53.59366067	0
6	88.7265625	40.67222541	0.600866079	1.123491692	1.178929766	11.4687196	14.26957284	252.5673058	0
7	93.5703125	46.69811352	0.53190485	0.416721117	1.636287625	14.54507425	10.6217484	131.3940043	0
8	119.484375	48.76505927	0.03146022	-0.112167573	0.99916388	9.279612239	19.20623018	479.7565669	0
9	130.3828125	39.84405561	-0.158322759	0.389540448	1.220735786	14.37894124	13.53945602	198.2364565	0
10	107.25	52.62707834	0.452688025	0.170347382	2.331939799	14.48685311	9.001004441	107.9725056	0
11	107.2578125	39.49648839	0.465881961	1.162877124	4.079431438	24.98041798	7.397079948	57.78473789	0
12	142.078125	45.28807262	-0.320328426	0.283952506	5.376254181	29.00989748	6.076265849	37.83139335	0
13	133.2578125	44.05824378	-0.081059862	0.115361506	1.632107023	12.00780568	11.97206663	195.5434476	0
14	134.9609375	49.55432662	-0.135303833	-0.080469602	10.69648829	41.34204361	3.893934139	14.13120625	0
15	117.9453125	45.50657724	0.325437564	0.661459458	2.836120401	23.11834971	8.943211912	82.47559187	0
16	138.1796875	51.5244835	-0.031852329	0.046797173	6.330267559	31.57634673	5.155939859	26.14331017	0
17	114.3671875	51.94571552	-0.094498904	-0.287984087	2.738294314	17.19189079	9.050612454	96.61190318	0
18	109.640625	49.01765217	0.13763583	-0.256699775	1.508361204	12.07290134	13.36792556	223.4384192	0
19	100.8515625	51.74352161	0.393836792	-0.011240741	2.841137124	21.63577754	8.302241891	71.58436903	0
20	136.09375	51.69100464	-0.045908926	-0.271816393	9.342809365	38.09639955	4.345438138	18.67364854	0
21	99.3671875	41.57220208	1.547196967	4.154106043	27.55518395	61.71901588	2.20880796	3.662680136	1
22	100.890625	51.89039446	0.627486528	-0.026497802	3.883779264	23.04526673	6.953167635	52.27944038	0
23	105.4453125	41.13996851	0.142653801	0.320419676	3.551839465	20.75501684	7.739552295	68.51977061	0
24	95.8671875	42.05992212	0.326386917	0.803501794	1.83277592	12.24896949	11.249331	177.2307712	0
25	117.3671875	53.90861351	0.257953441	-0.405049077	6.018394649	24.76612335	4.807783224	25.52261561	0
26	106.6484375	56.36718209	0.378355072	-0.266371607	2.43645485	18.40537062	9.378659682	96.86022536	0
27	112.71875	50.3012701	0.279390953	-0.129010712	8.281772575	37.81001224	4.691826852	21.27620977	0
28	130.8515625	52.43285734	0.142596727	0.018885442	2.64632107	15.65443599	9.464164025	115.6731586	0
29	119.4375	52.87481531	-0.002549267	-0.460360287	2.365384615	16.49803188	9.008351898	94.75565692	0
30	123.2109375	51.07801208	0.179376819	-0.17728516	2.107023411	16.92177312	10.08033334	112.5585913	0
31	102.6171875	49.69235371	0.230438984	0.193325371	1.489130435	16.00441146	12.64653474	171.8329021	0
32	110.109375	41.31816988	0.094860398	0.68311261	1.010033445	13.02627521	14.66651082	231.2041363	0
33	99.9140625	43.91949797	0.475728501	0.781486196	0.619565217	9.440975862	20.1066391	475.680218	0

機械学習プロセス


課題設定
データ取得
データ前処理
手法選択
モデル学習
モデル評価
・分析から得たいものを決定
・ログ
・アンケート
・WebAPI
etc..
・欠損値処理
・特徴量選択
・次元削除
etc..
・SVM
・決定木
・k-NN
etc..
・手法選択で選択をしたモデルを使用し学習させる
・正解率
・適合率
・再現率
・汎化性能の評価
etc..

（復習）機械学習のアルゴリズム

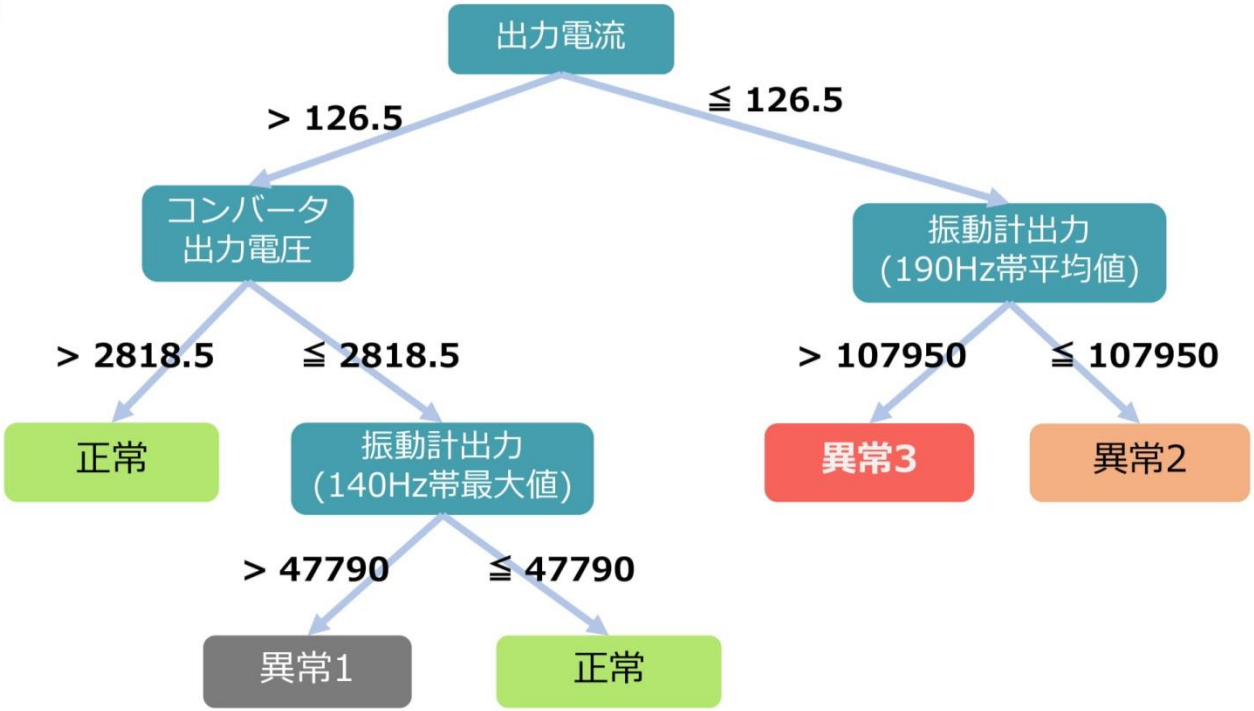
- 線形回帰
- K近傍法
- 決定木
- Gradient Boosting Machine
- ロジスティック回帰
- Support Vector Machine
- Neural Network
- Deep Learning　　etc…

今回は、種々の機械学習のアルゴリズムで機械学習を実施し結果を比較する。

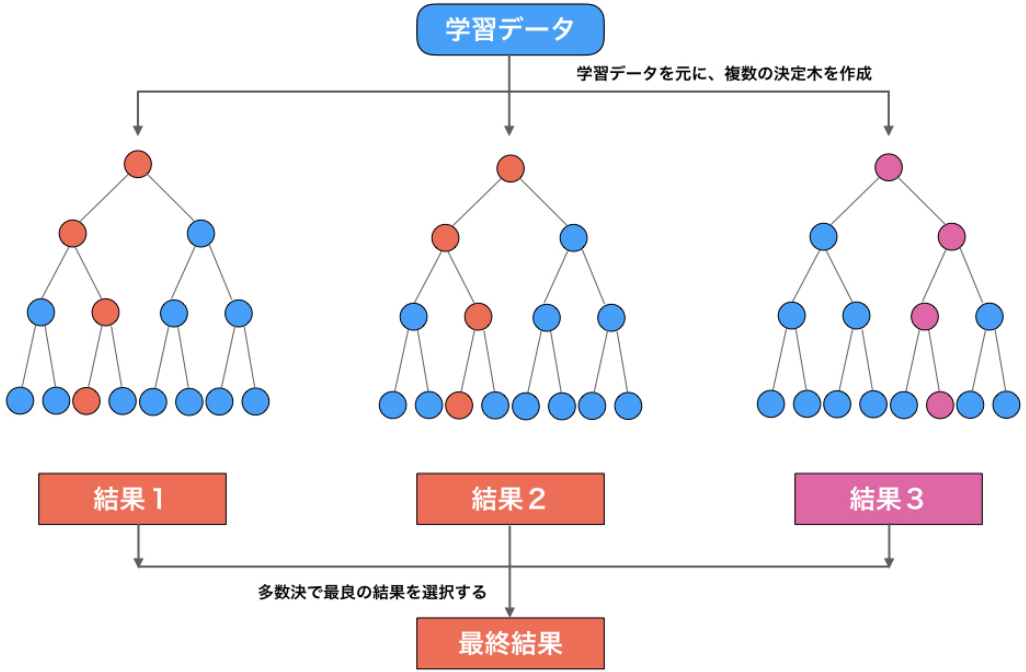
機械学習はアルゴリズムの知識が全くなくても使えるが、各アルゴリズムの概要だけ解説する。

（復習）決定木について

決定木とは



ランダムフォレスト



＜特徴＞
- 比較的精度が高い
- 計算時間は少し長め
- 基本的に過学習*を起こさないと言われている。

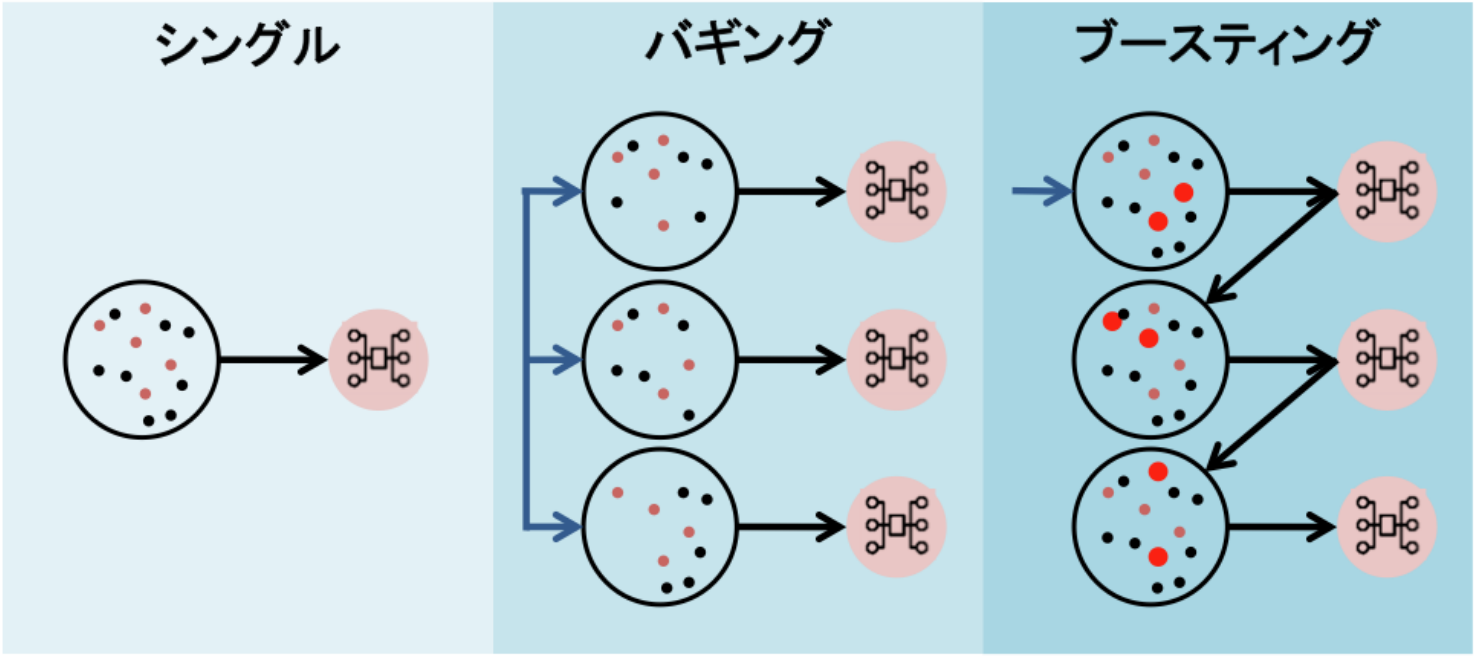
（アルゴリズムのスクリーニングにはRFを入れることを勧めます）
★私の一押しのアルゴリズムです。

*過学習については、3回目以降で説明します。

あまり精度が高くない**分類器**（弱分類器）を複数組み合わせる**アンサンブル学習**の一種

ランダムフォレストは決定木（弱分類器）を**バギング**という方法でアンサンブル学習させる

アンサンブル学習

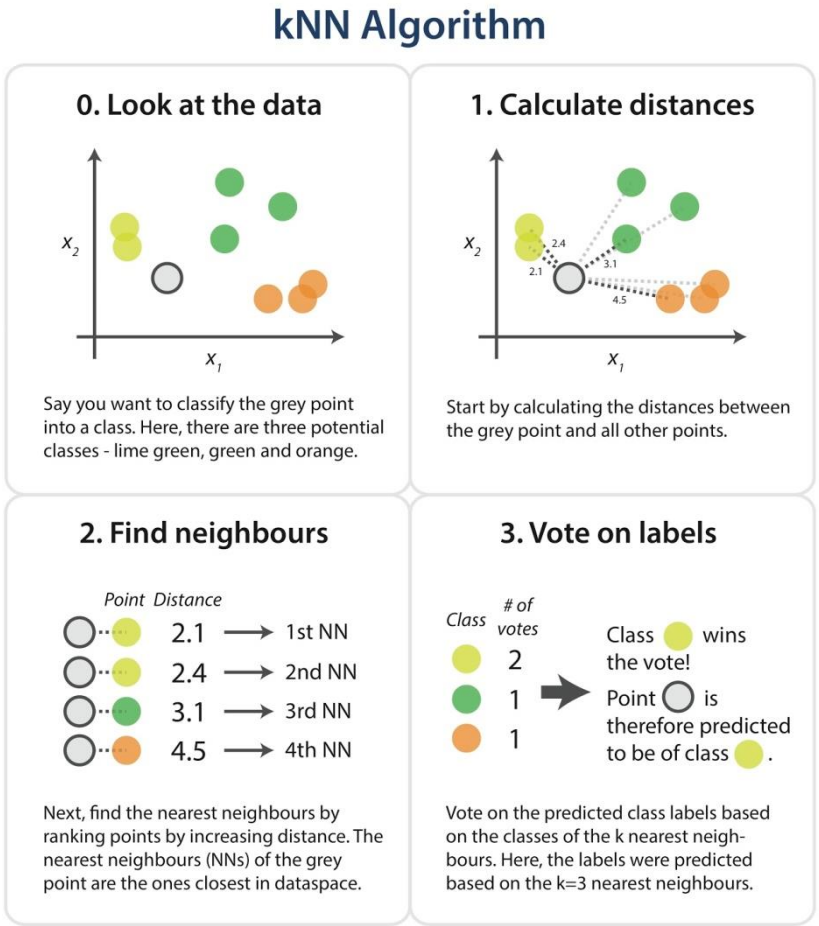


（例）決定木

ランダムフォレスト

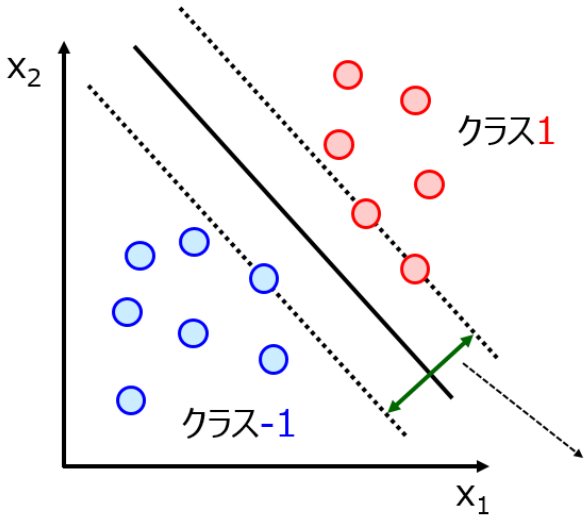
AdaBoost
XGBoost
LightGBM

K-近傍法(k-NN)



＜特徴＞
- 精度はほどほど(問題による)
- 計算時間は短い
- 複雑な境界を学習することができる

サポートベクターマシーン（SVM)

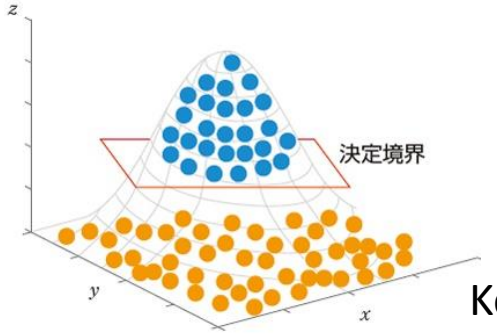


マージンを最大化するように
判別関数を決める！

$$f(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$
$$= \mathbf{xw} + b$$

マージン $= \dfrac{2}{\|\mathbf{w}\|} = \dfrac{2}{\sqrt{w_1^2 + w_2^2}}$

(点と直線との距離で計算)



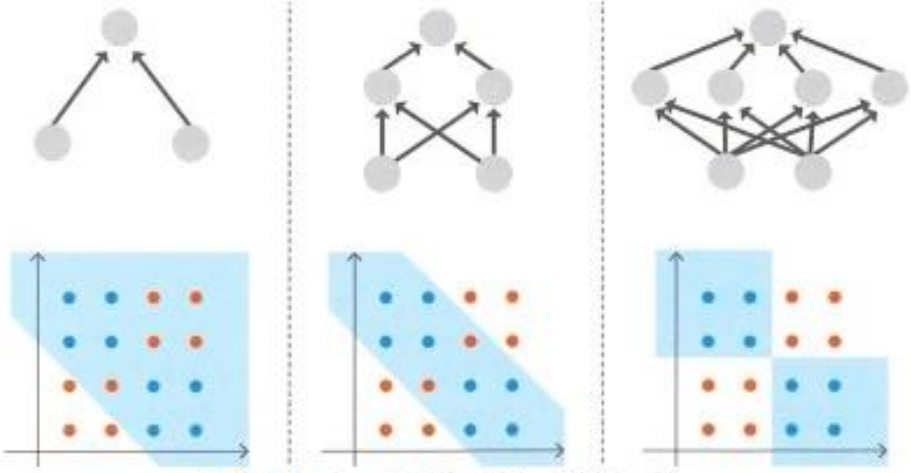
Kernel法

＜特徴＞
- 精度はほどほど（問題による）
- 計算時間は短い
- 複雑なモデルでは過学習を起こす場合がある。

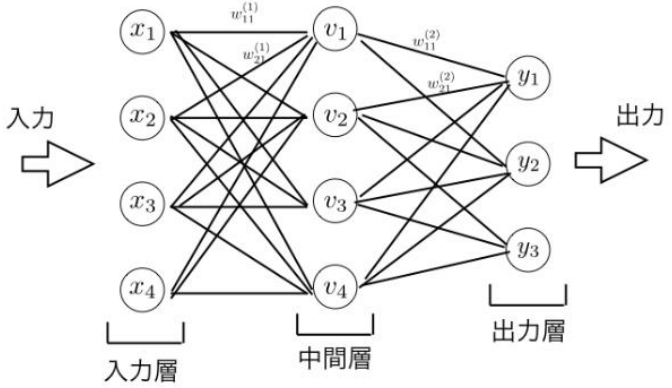
（色々なタイプがあるので試してて、
ツボにはまると精度が高くでることもある）

基本的なSVMは直線で分けるが、曲線や多次元で分ける方法等種々のSVMがある

ニューラルネットワーク

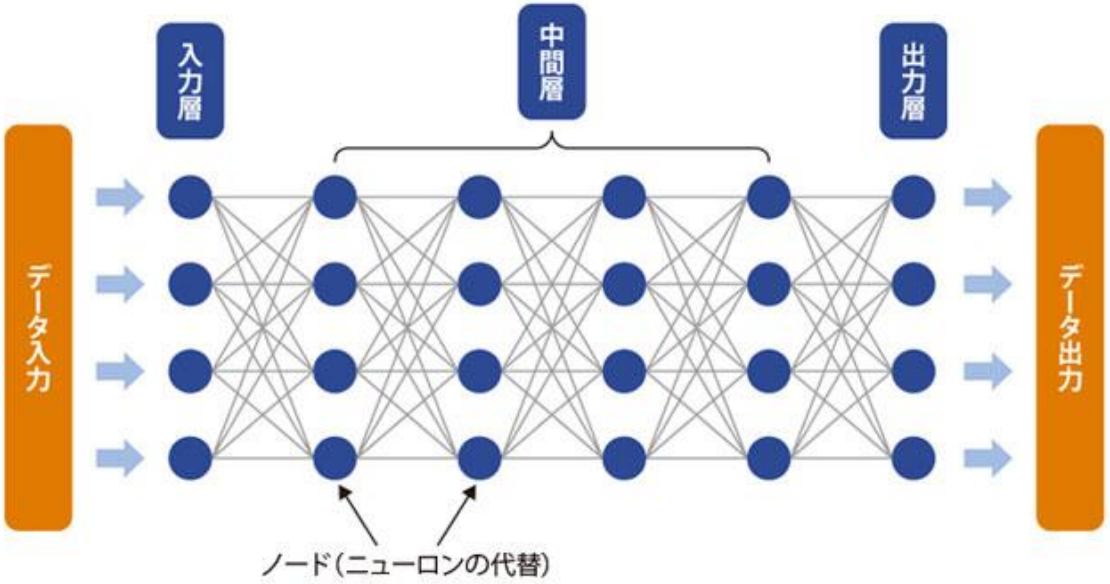


▲図2.8.8　ニューラルネットワークのイメージ



＜特徴＞
- 精度は高い
- 計算時間は長め
- 複雑なモデル（例中間層を増やす）では過学習を起こす場合がある。

ディープラーニング



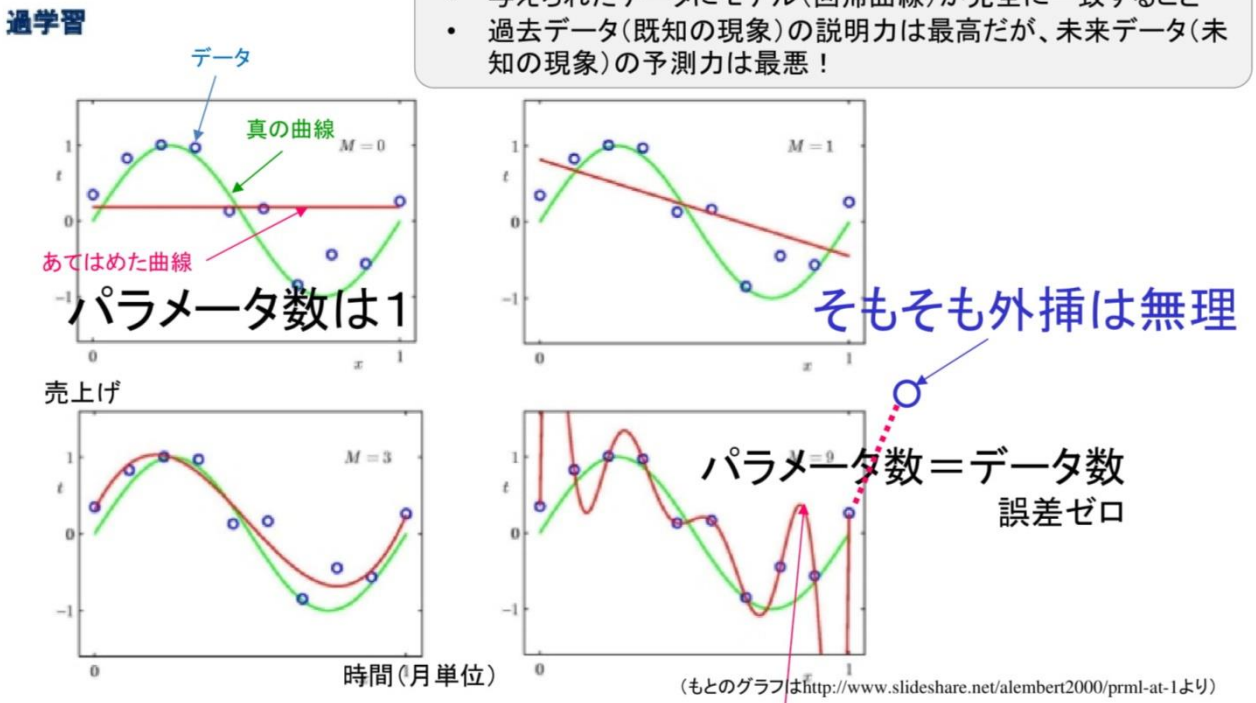
NNを多層にしたもの。
パターン認識に向いている。

＜特徴＞
・精度はまちまち
（データとの相性がある）
・計算時間は（やや）長め
・画像の判別には向いている
・オプションが多く、
ブラックボックス部分が多い

過学習

表現能力の高いモデルには過学習（over fitting）が避けられない

- 与えられたデータにモデル（回帰曲線）が完全に一致すること
- 過去データ（既知の現象）の説明力は最高だが、未来データ（未知の現象）の予測力は最悪！



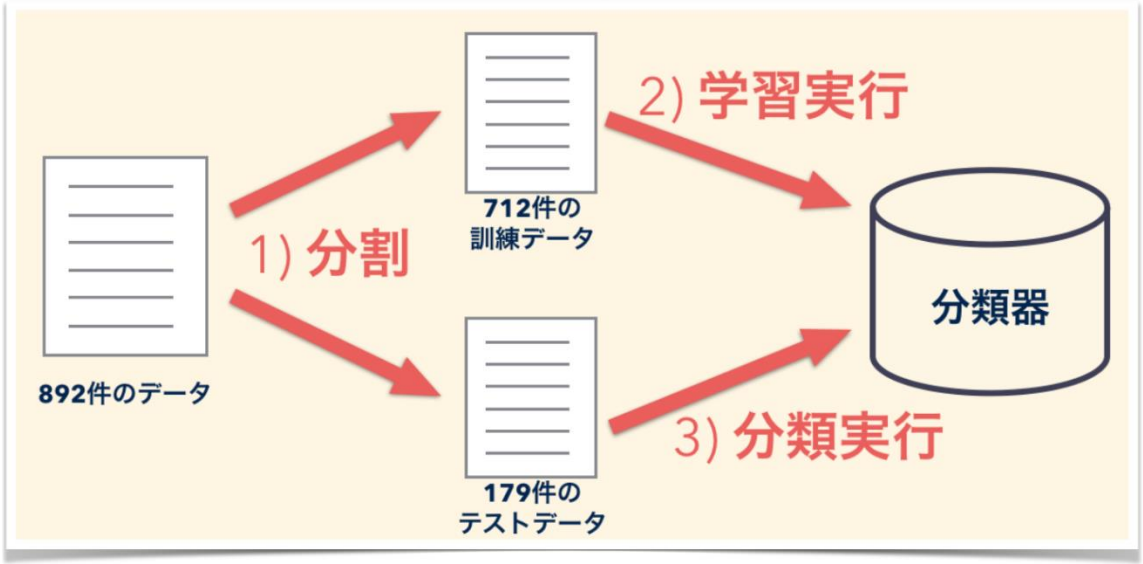
（もとのグラフはhttp://www.slideshare.net/alembert2000/prml-at-1より）

回帰曲線で起こる過学習は、モデルを複雑にすると、他の分類器でも起きうる。

過学習と実際の精度検証

本勉強会では、2つの手法を取り扱います。

（1）教師（訓練）データとテストデータに**分割**



（2）交差検証（Cross Validation） → 次回の勉強会で取り扱います。

機械学習デモ＋実習

3. データ分割による検証
ROC曲線 と AUC

Result History | Random Forest Model (Random Forest) | ExampleSet (Apply Model) | PerformanceVector (Performance) | ExampleSet (Multiply (2))

Performance | Description | Annotations

Criterion
- accuracy
- kappa
- AUC (optimistic)
- AUC
- AUC (pessimistic)
- precision
- positive predictive value
- negative predictive value

AUC: 0.966 (positive class: true)

ROC | ROC (Thresholds)

True positive

False positive

ROC = Receiver Operating Characteristic Curve

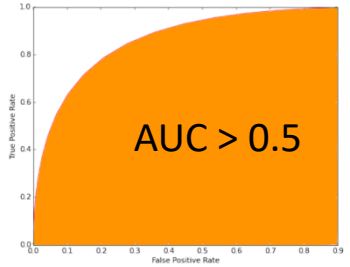
AUC = Area Under Curve (ROC曲線の下のエリア）

AUC	効果の評価
0.8以上	効果がすごくいい
0.7-0.8	効果があります
0.6-0.7	効果が多少あります
0.5-0.6	効果がありません

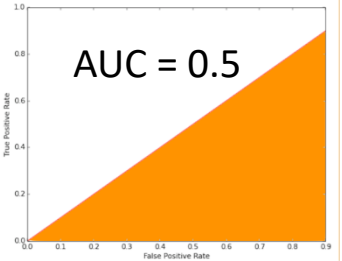
＜今回の結果＞
AUC (optimistic): 0.966 (positive class: true)
AUC: 0.966 (positive class: true)
AUC (pessimistic): 0.966 (positive class: true)

ROC曲線(Receiver Operatorating Characteristic curve、受信者動作特性曲線)は、もともとレーダーシステムの通信工学理論として開発されたものであり、レーダー信号のノイズの中から敵機の存在を検出するための方法として開発された方法です

分類器の効果



効果がある分類器



効果がない分類器

4. 分類問題の評価指数（2値問題）

混合行列(Confusion Matrix)とTP, FP, FN, TN

	実際は正 (Positive)	実際は負 (Negative)
予測が正 (Positive)	**TP** True Positive	**FP** False Positive 第1種の誤り
予測が負 (Negative)	**FN** Flase Negative 第2種の誤り	**TN** True Negative

適合率と再現率はトレードオフの関係
両者のバランスを評価するのがF1値

参考：
https://qiita.com/FukuharaYohei/items/be89a99c53586fa4e2e4

指標	意味	式
正解率(Accuracy)	全予測 正答率	$\frac{TP + TN}{TP + FP + FN + TN}$
適合率(Precision)	正予測 の正答率	$\frac{TP}{TP + FP}$
再現率(Recall)	正に対する 正答率	$\frac{TP}{TP + FN}$
特異率(Specificity)	負に対する 正答率	$\frac{TN}{FP + TN}$
F値(F-measure)	適合率と 再現率の 調和平均	$\frac{2 \times 適合率 \times 再現率}{適合率 + 再現率}$