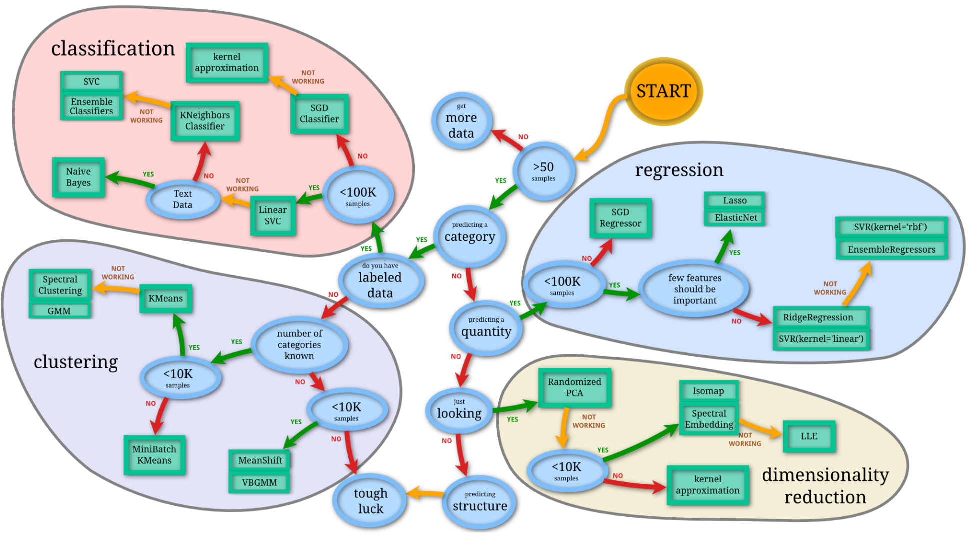
Pattern Recognition

Bertrand Thirion and John Ashburner


START
>50 samples
get more data
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
<100K samples
Linear SVC
Text Data
Naive Bayes
KNeighbors Classifier
SVC
Ensemble Classifiers
SGD Classifier
kernel approximation
clustering
number of categories known
<10K samples
KMeans
Spectral Clustering
GMM
MiniBatch KMeans
MeanShift
VBGMM
regression
few features should be important
SGD Regressor
Lasso
ElasticNet
RidgeRegression
SVR(kernel='linear')
SVR(kernel='rbf')
EnsembleRegressors
dimensionality reduction
Randomized PCA
Isomap
Spectral Embedding
LLE
kernel approximation
YES
NO
NOT WORKING

Some key concepts

supervised learning: The data comes with additional attributes that we want to predict $\Longrightarrow$ classification and regression.

unsupervised learning: No target values.

- Discover groups of similar examples within the data (clustering).
- Determine the distribution of data within the input space (density estimation).
- Project the data down to two or three dimensions for visualization.

General supervised learning setting

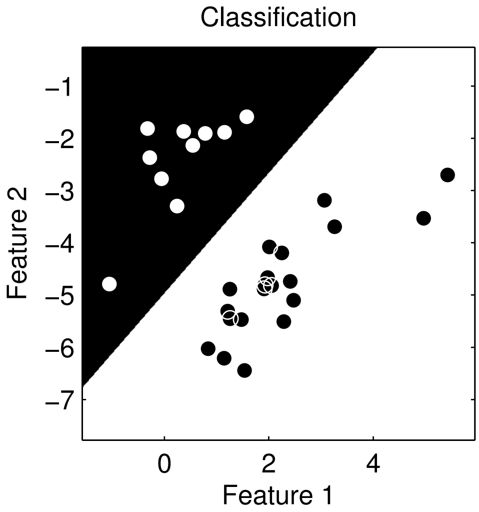
We have a training dataset of n observations, each consisting of an input $\mathbf{x}_i$ and a target y_i.
Each input, $\mathbf{x}_i$, consists of a vector of p features.

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, .., n\}$$

The aim is to predict the target for a new input $\mathbf{x}_*$.

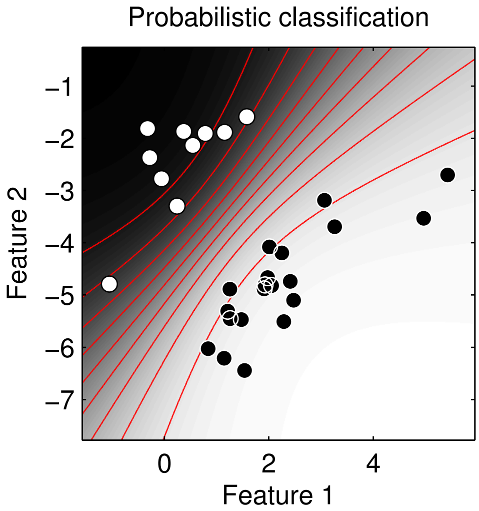
Classification

Targets (**y**) are categorical labels.
Train with $\mathcal{D}$ and use result to make best guess of y_* given $\mathbf{x}_*$.



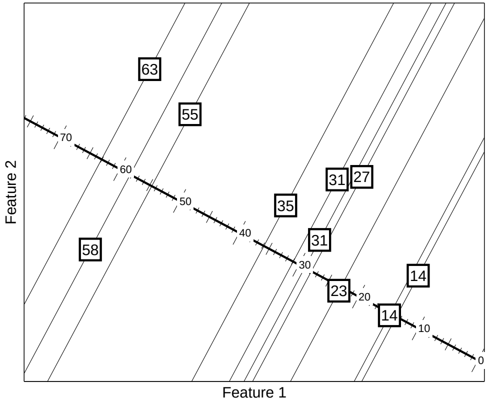
Probabilistic classification

Targets (**y**) are categorical labels.
Train with $\mathcal{D}$ and compute $P(y_* = k|\mathbf{x}_*, \mathcal{D})$.



Regression

Targets ($\mathbf{y}$) are continuous real variables.
Train with $\mathcal{D}$ and compute $p(y_*|\mathbf{x}_*, \mathcal{D})$.



Many other settings

- **Multi-class classification** when there are more than two possible categories.
- **Ordinal regression** for classification when there is some ordering of the categories.
  Chu, Wei, and Zoubin Ghahramani. "Gaussian processes for ordinal regression." In Journal of Machine Learning Research, pp. 1019-1041. 2005.
- **Multi-task learning** when there are multiple targets to predict, which may be related.
- etc

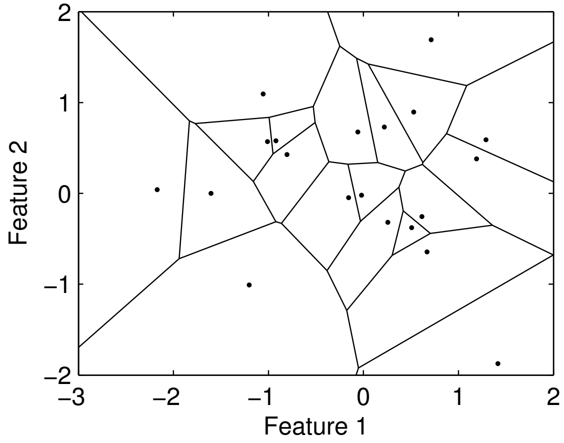
Multi-Class classification

- **Multinomial Logistic regression** Theoretically optimal. Expensive optimization.
- **One-versus-all classification** [SVMs] Among several hyperplane, choose the one with maximal margin.
  $\Longrightarrow$ recommended
- **One-versus-one classification** Vote across each pair of class. Expensive, not optimal.

Curse of dimensionality

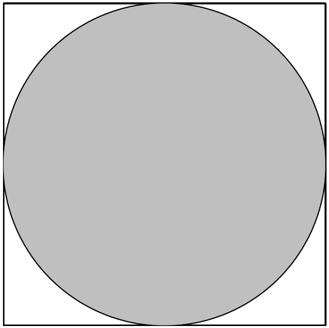
Large p, small n.

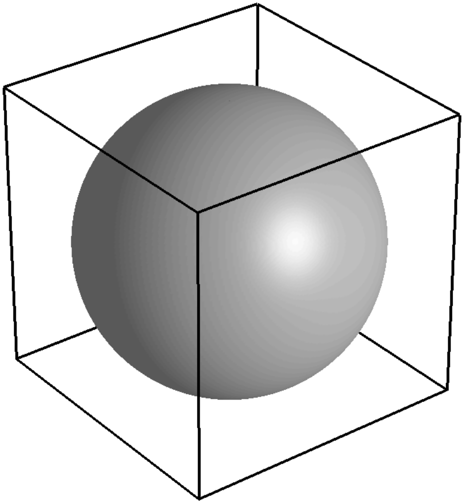
Nearest-neighbour classification



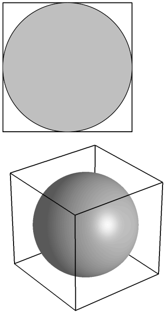
- Not nice smooth separations.
- Lots of sharp corners.
- May be improved with *K-nearest neighbours*.

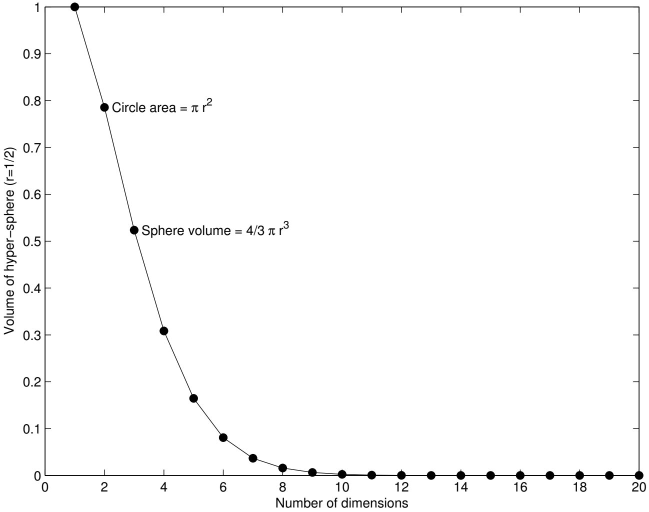
Behaviour changes in high-dimensions





Behaviour changes in high-dimensions





Occam's razor

"Everything should be kept as simple as possible, but no simpler."

— Einstein (allegedly)

- Complex models (with many estimated parameters) usually explain training data better than simpler models.
- Simpler models often generalise better to new data than nore complex models.

Need to find the model with the optimal bias/variance tradeoff.

Bayesian model selection

Real Bayesians don't cross-validate (except when they need to).

$$P(M|\mathcal{D}) = \frac{p(\mathcal{D}|M)P(M)}{p(\mathcal{D})}$$

The *Bayes factor* allows the plausibility of two models (M_1 and M_2) to be compared:

$$K = \frac{p(\mathcal{D}|M_1)}{p(\mathcal{D}|M_2)} = \frac{\int_{\theta_{M_1}} p(\mathcal{D}|\theta_{M_1}, M_1)p(\theta_{M_1}|M_1)d\theta_{M_1}}{\int_{\theta_{M_2}} p(\mathcal{D}|\theta_{M_2}, M_2)p(\theta_{M_2}|M_2)d\theta_{M_2}}$$

This is usually too costly in practice, so approximations are used.

Model selection

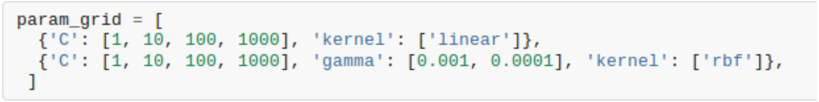
Some approximations/alternatives to the Bayesian approach:

- **Laplace approximations**: find the MAP/ML solution and use a Gaussian approximation to the parameter uncertainty.
- **Minimum Message Length** (MML): an information theoretic approach.
- **Minimum Description Length** (MDL): an information theoretic approach based on how well the model compresses the data.
- **Akaike Information Criterion** (AIC): $-2 \log p(\mathcal{D}|\theta) + 2k$, where k is the number of estimated parameters.
- **Bayesian Information Criterion** (BIC): $-2 \log p(\mathcal{D}|\theta) + k \log q$, where q is the number of observations.

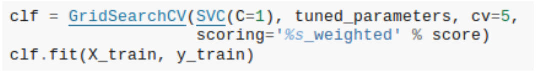
Model selection by nested cross-validation

Inner cross-validation loop used to evaluate model's performance on a pre-defined grid of parameters and retain the best one.

- Safe, but costly.
- Supported by some libraries (e.g. scikit-learn).



```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
 ]
```



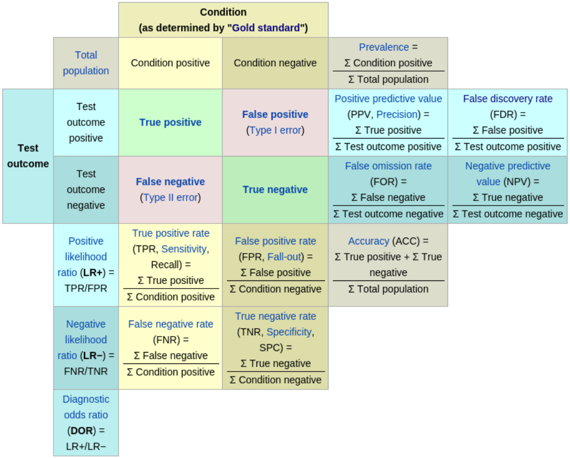
```
clf = GridSearchCV(SVC(C=1), tuned_parameters, cv=5,
                   scoring='%s_weighted' % score)
clf.fit(X_train, y_train)
```

- Some estimators have path model, hence allow faster evaluation (e.g. LASSO).
- Randomized techniques also exist, sometimes more efficient.
- **Caveat:** Inner cross-validation loop $\neq$ outer cross-validation loop for parameter evaluation.

Accuracy measures for regression

- **Root-mean squared error** for point predictions.
- **Correlation coefficient** for point predictions.
- **Log predictive probability** can be used for probabilistic predictions.
- **Expected loss/risk** for point predictions for decision making.

Accuracy measures for binary classification



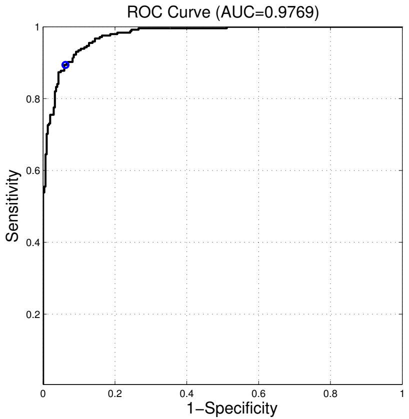
		Condition (as determined by "Gold standard")			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV, Precision) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$
	Test outcome negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Positive likelihood ratio (**LR+**) = TPR/FPR	True positive rate (TPR, Sensitivity, Recall) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR, Fall-out) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
	Negative likelihood ratio (**LR−**) = FNR/TNR	False negative rate (FNR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR, Specificity, SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$		
	Diagnostic odds ratio (**DOR**) = LR+/LR−				

Wikipedia contributors, "Sensitivity and specificity," Wikipedia, The Free Encyclopedia, `http://en.wikipedia.org/w/index.php?title=Sensitivity_and_specificity&oldid=655245669` (accessed April 9, 2015).

Accuracy measures from ROC curve

The **Receiver operating characteristic** (ROC) curve is a plot of *true-positive rate* (sensitivity) versus *false-positive rate* (1-specificity) over the full range of possible thresholds.

The **area under the curve** (AUC) is the integral under the ROC curve.



Log predictive probability

Some data are more easily classified than others.
Probabilistic classifiers provide a level of confidence for each prediction.

$$p(y_* | \mathbf{x}_*, \mathbf{y}, \mathbf{X}, \theta)$$

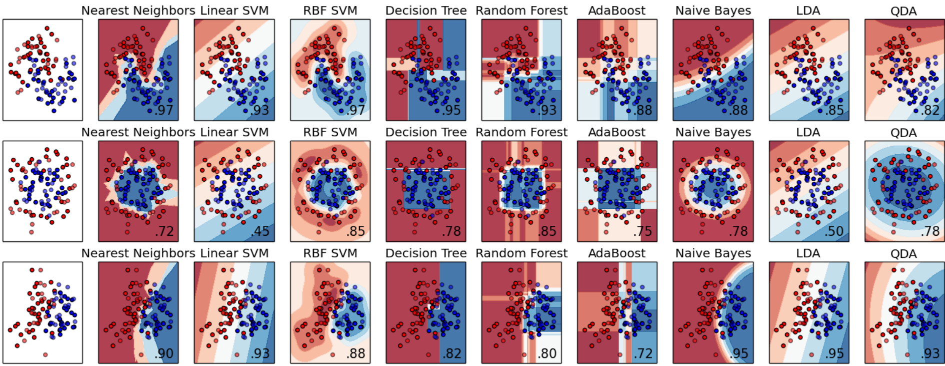
Quality of predictions can be assessed using the **test log predictive probability**:

$$\frac{1}{m} \sum_{i=1}^{m} \log_2 p(y_{*i} = t_i | \mathbf{x}_{*i}, \mathbf{y}, \mathbf{X}, \theta)$$

After subtracting the baseline measure, this shows the average bits of information given by the model.

Rasmussen & Williams. "Gaussian Processes for Machine Learning", MIT Press (2006).
http://www.gaussianprocess.org/gpml/

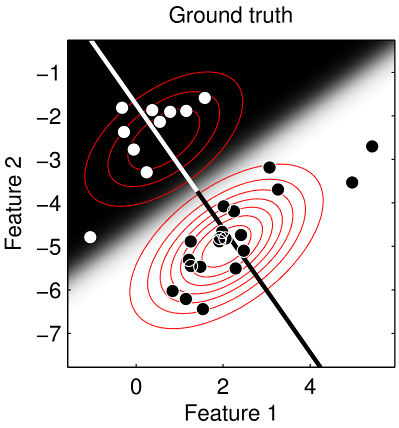
Overview of classification tools



Only one rule: No tool wins in all situations.

Generative models for classification

$$P(y=k|\mathbf{x}) = \frac{P(y=k)p(\mathbf{x}|y=k)}{\sum_j P(y=j)p(\mathbf{x}|y=j)}$$

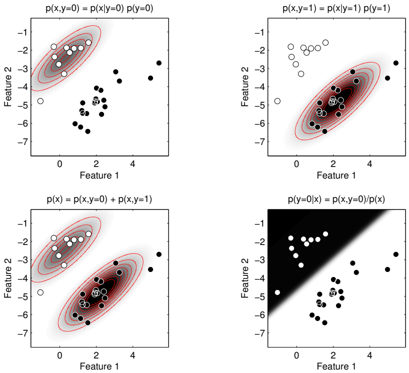


Linear discriminant analysis

$$P(y=k|\mathbf{x}) = \frac{P(y=k)p(\mathbf{x}|y=k)}{\sum_j P(y=j)p(\mathbf{x}|y=j)}$$

Assumes:

$$P(\mathbf{x}|y=k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$



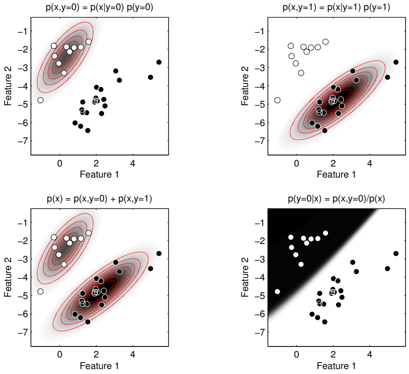
Model has $2p + p(p-1)$ parameters to estimate (two means and a single covariance).
Number of observations is pn (size of inputs).

Quadratic discriminant analysis

$$P(y=k|\mathbf{x}) = \frac{P(y=k)p(\mathbf{x}|y=k)}{\sum_j P(y=j)p(\mathbf{x}|y=j)}$$

Assumes different covariances:

$$P(\mathbf{x}|y=k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$



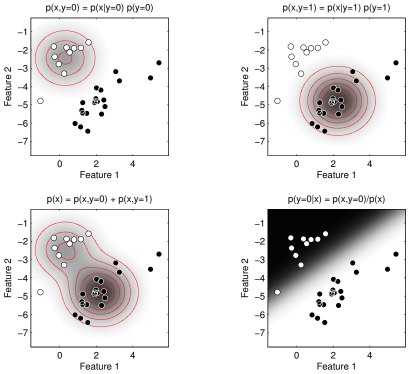
Model has $2p + 2p(p-1)$ parameters to estimate (two means and two covariances).
Number of observations is pn.

Naive Bayes

$$P(y=k|\mathbf{x}) = \frac{P(y=k)p(\mathbf{x}|y=k)}{\sum_j P(y=j)p(\mathbf{x}|y=j)}$$

Assumes that features are independent:

$$p(\mathbf{x}|y=k) = \prod_i p(x_i|y=k)$$



Model has variable number of parameters to estimate, but the above example has $3p$.
Number of observations is pn.

Linear regression: maximum likelihood

A simple way to do regression is by:

$$f(\mathbf{x}_*) = \mathbf{w}^T \mathbf{x}_*$$

Assuming Gaussian noise on $\mathbf{y}$, the ML estimate of $\mathbf{w}$ is by:

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

where

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_n \end{pmatrix}^T, \text{ and } \mathbf{y} = \begin{pmatrix} y_1 & y_2 & \ldots y_n \end{pmatrix}^T$$

Model has p parameters to estimate.
Number of observations is n (number of targets).
Usually needs dimensionality reduction, with (eg) SVD.

Linear regression: maximum posterior

We may have prior knowledge about various distributions:

$$\begin{aligned} p(y_*|\mathbf{x}_*, \mathbf{w}) =& \mathcal{N}(\mathbf{w}^T\mathbf{x}_*, \sigma^2) \\ p(\mathbf{w}) =& \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_0) \end{aligned}$$

Therefore,

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \mathcal{N}(\sigma^{-2}\mathbf{B}^{-1}\mathbf{X}^T\mathbf{y}, \mathbf{B}^{-1}), \text{ where } \mathbf{B} = \sigma^{-2}\mathbf{X}^T\mathbf{X} + \mathbf{\Sigma}_0^{-1}$$

Maximum a posteriori (MAP) estimate of $\mathbf{w}$ is by:

$$\hat{\mathbf{w}} = \sigma^{-2}\mathbf{B}^{-1}\mathbf{X}^T\mathbf{y}, \text{ where } \mathbf{B} = \sigma^{-2}\mathbf{X}^T\mathbf{X} + \mathbf{\Sigma}_0^{-1}$$

Linear regression: Bayesian

We may have prior knowledge about various distributions:

$$p(y_*|\mathbf{x}_*, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T\mathbf{x}_*, \sigma^2)$$
$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_0)$$

Therefore,

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \mathcal{N}(\sigma^{-2}\mathbf{B}^{-1}\mathbf{X}^T\mathbf{y}, \mathbf{B}^{-1}), \text{ where } \mathbf{B} = \sigma^{-2}\mathbf{X}^T\mathbf{X} + \mathbf{\Sigma}_0^{-1}$$

Predictions are made by integrating out the uncertainty of the weights, rather than estimating them:

$$p(y_*|\mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \int_{\mathbf{w}} p(y_*|\mathbf{x}_*, \mathbf{w})p(\mathbf{w}|\mathbf{y}, \mathbf{X})d\mathbf{w}$$
$$= \mathcal{N}(\sigma^{-2}\mathbf{x}_*^T\mathbf{B}^{-1}\mathbf{X}^T\mathbf{y}, \mathbf{x}_*^T\mathbf{B}^{-1}\mathbf{x}_*)$$

Estimated parameters may be σ^2, and parameters encoding $\mathbf{\Sigma}_0$.

Kernel methods: Woodbury matrix identity

$$\begin{aligned}\mathbf{B}^{-1} &= \left(\sigma^{-2}\mathbf{X}^T\mathbf{X} + \mathbf{\Sigma}_0^{-1}\right)^{-1} && \text{invert a } p \times p \text{ matrix}\\ &= \mathbf{\Sigma}_0 - \mathbf{\Sigma}_0\mathbf{X}^T(\mathbf{I}\sigma^2 + \mathbf{X}\mathbf{\Sigma}_0\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{\Sigma}_0 && \text{invert an } n \times n \text{ matrix}\end{aligned}$$

Wikipedia contributors, "Woodbury matrix identity," Wikipedia, The Free Encyclopedia, `http://en.wikipedia.org/w/index.php?title=Woodbury_matrix_identity&oldid=638370219` (accessed April 1, 2015).
$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}$.

Kernel methods: Gaussian process regression

The predicted distribution is:

$$p(y_* | \mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \mathcal{N}(\mathbf{k}^T \mathbf{C}^{-1} \mathbf{y}, c - \mathbf{k}^T \mathbf{C}^{-1} \mathbf{k})$$

where:

$$\begin{aligned} \mathbf{C} &= \mathbf{X} \mathbf{\Sigma}_0 \mathbf{X}^T + \mathbf{I}\sigma^2 \\ \mathbf{k} &= \mathbf{X} \mathbf{\Sigma}_0 \mathbf{x}_* \\ c &= \mathbf{x}_*^T \mathbf{\Sigma}_0 \mathbf{x}_* + \sigma^2 \end{aligned}$$

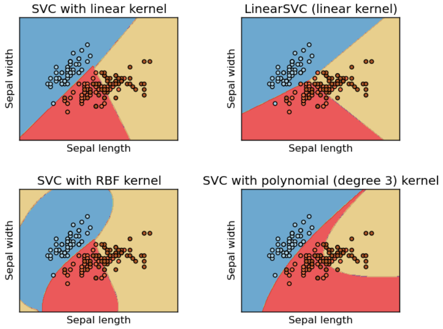
Kernel methods: nonlinear methods

Sometimes, we want alternatives to $\mathbf{C} = \mathbf{X}\mathbf{\Sigma}_0\mathbf{X}^T + \mathbf{I}\sigma^2$.
Nonlinearity is achieved by replacing the matrix $\mathbf{K} = \mathbf{X}\mathbf{\Sigma}_0\mathbf{X}^T$ with some function of the data that gives a positive definite matrix encoding similarities.
eg

$$k(\mathbf{x}_i, \mathbf{x}_j) = \theta_1 + \theta_2 \mathbf{x}_i \cdot \mathbf{x_j} + \theta_3 \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x_j}||^2}{2\theta_4^2}\right)$$

Hyper-parameters θ_1 to θ_4 can be optimised in a number of ways.

Kernel methods: nonlinear methods



Non-linear methods are useful in low-dimension to adapt the shape of decision boundaries.

For large p, small n problems, nonlinear methods do not seem to help much.
Nonlinearity also reduces interpretability.

Probabilistic discriminative models

Regression

Continuous targets:

$$y \in \mathcal{R}$$

Usually assume a Gaussian distribution:

$$p(y|\mathbf{x}, \mathbf{w}) = \mathcal{N}(f(\mathbf{x}, \mathbf{w}), \sigma^2)$$

where σ^2 is a variance.

Binary Classification

Categorical targets:

$$y \in \{0, 1\}$$

Usually assume a binomial distribution:

$$p(y|\mathbf{x}, \mathbf{w}) = \sigma(f(\mathbf{x}, \mathbf{w}))^y (1 - \sigma(f(\mathbf{x}, \mathbf{w})))^{1-y}$$

where σ is a squashing function.

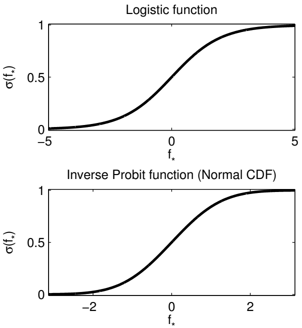
Probabilistic discriminative models

For binary classification:

$$p(y_* = 1|\mathbf{x}_*, \mathbf{w}) = \sigma(f(\mathbf{x}_*, \mathbf{w}))$$

where σ is some squashing function, eg:

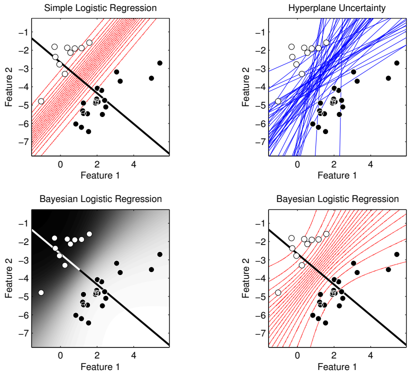
- Logistic sigmoid function (inverse of Logit).
- Normal CDF (inverse of Probit).



Probabilistic discriminative models

Integrating over the uncertainty of the separating hyperplane allows probabilistic predictions further from the training data. This is not usually done for methods such as the relevance-vector machine (RVM).

Rasmussen, Carl Edward, and Joaquin Quinonero-Candela. "Healing the relevance vector machine through augmentation." In Proceedings of the 22nd international conference on Machine learning, pp. 689-696. ACM, 2005.



Probabilistic discriminative models

Making probabilistic predictions involves:

1. Computing the distribution of a latent variable corresponding to the test data (cf regression):

$$p(f_*|\mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \int_{\mathbf{f}} p(f_*|\mathbf{x}_*, \mathbf{f})p(\mathbf{f}|\mathbf{y}, \mathbf{X})d\mathbf{f}$$

2. Using this distribution to give a probabilistic prediction:

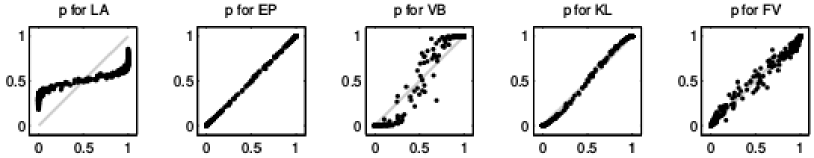
$$P(y_* = 1|\mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \int_{f_*} \sigma(f_*)p(f_*|\mathbf{x}_*, \mathbf{y}, \mathbf{X})df_*$$

Unfortunately, these integrals are analytically intractable, so approximations are needed.

Probabilistic discriminative models

Approximate methods for probabilistic classification include:

- **The Laplace Approximation** (LA). Fastest, but less accurate.
- **Expectation Propagation** (EP). More accurate than the Laplace approximation, but slightly slower.
- **MCMC** methods. The "gold standard", but very slow to draw lots of random samples.



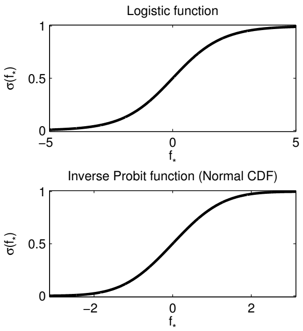
Nickisch, Hannes, and Carl Edward Rasmussen. "Approximations for Binary Gaussian Process Classification." Journal of Machine Learning Research 9 (2008): 2035-2078.

Discriminative models for classification

$$t = \sigma(f(\mathbf{x}_*))$$

where σ is some squashing function, eg:

- Logistic function (inverse of Logit).
- Normal CDF (inverse of Probit).
- Hinge loss (support vector machines)



Discriminative models for classification: convexity

In practice, the hinge and logistic losses yield a convex estimation problem and are preferred.

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \mathcal{L}(y_i, \mathbf{X_i}, \mathbf{w}) + \lambda \mathcal{R}(\mathbf{w})$$

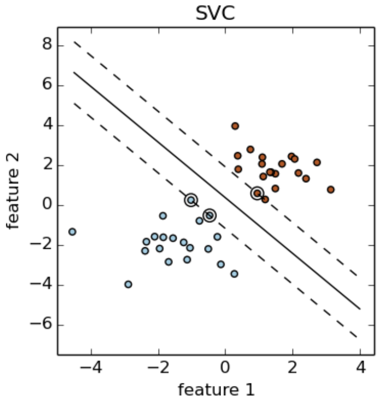
(M-estimators framework)

- $\mathcal{L}$ is the loss function (hinge, logistic, quadratic...)
- $\mathcal{R}$ is the regularizer (typically a norm on $\mathbf{w}$)
- $\lambda > 0$ balances the two terms

$\mathcal{L}$ and $\mathcal{R}$ convex $\rightarrow$ unique minimizer (SVMs, ℓ_2-logistic, ℓ_1-logistic).

Support vector classification

SVMs are reasonably fast, accurate and easy to tune ($C = 10^3$ is a good default, no dramatic failure).
Multi-class: One-versus-one, One-versus all.



Ensemble learning

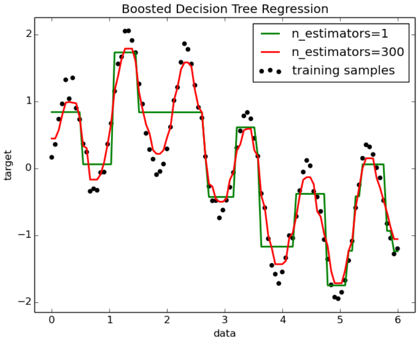
Combining predictions from weak learners.

- **Bootstrap aggregating (bagging)**
  - Train several weak classifiers, with different models or randomly drawn subsets of the data.
  - Average their predictions with equal weight.
- **Boosting**
  - A family of approaches, where models are weighted according to their accuracy.
  - AdaBoost is popular, but has problems with target noise.
- **Bayesian model averaging**
  - Really a model selection method.
  - Relatively ineffective for combining models.
- **Bayesian model combination**
  - Shows promise.

Monteith, et al. "Turning Bayesian model averaging into Bayesian model combination." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

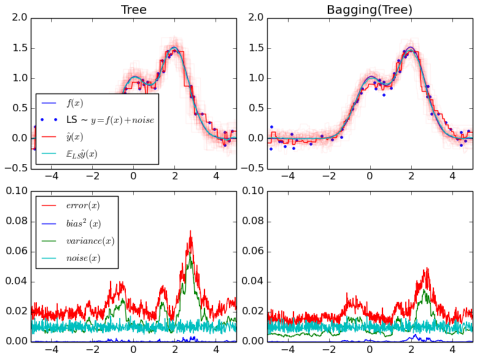
Boosting

Reduce sequentially the bias of the combined estimator.
Examples: AdaBoost, Gradient Tree Boosting, ...



Bagging

Build several estimators independently and average their predictions. Reduce the variance.
Examples: Bagging methods, Forests of randomized trees, ...



Free Books

- **The Elements of Statistical Learning: Data Mining, Inference, and Prediction** Trevor Hastie, Robert Tibshirani, Jerome Fried(2009)
  `http://statweb.stanford.edu/~tibs/ElemStatLearn/`
- **An Introduction to Statistical Learning with Applications in R** Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani (2013)
  `http://www-bcf.usc.edu/%7Egareth/ISL/`
- **Introduction to Machine Learning** Amnon Shashua (2008)
  `http://arxiv.org/pdf/0904.3664.pdf`

Free Books

- **Bayesian Reasoning and Machine Learning** David Barber (2014) http://www.cs.ucl.ac.uk/staff/d.barber/brml/
- **Gaussian Processes for Machine Learning** Carl Edward Rasmussen and Christopher K. I. Williams (2006) http://www.gaussianprocess.org/gpml/chapters/
- **Information Theory, Inference, and Learning Algorithms** David J.C. MacKay (2003) http://www.inference.phy.cam.ac.uk/itila/book.html

Web sites

- **Kernel Machines** http://www.kernel-machines.org/
- **The Gaussian Processes Web Site** includes links to software. http://www.gaussianprocess.org/
- **SVM - Support Vector Machines** includes links to software. http://www.support-vector-machines.org/
- **Pascal Video Lectures** http://videolectures.net/pascal

MATLAB tools

- **Spider** Object orientated environment for machine learning in MATLAB.
- **GPML** Gaussian processes for supervised learning.
- **Pronto** MATLAB ML tbx for neuroimaging. GUI. Implements many ML concepts. Continuity with SPM.

Python tools

- **Scikit-learn** Generic ML in Python. Complete, high-quality, well-documented, reference implementations.
- **Nilearn** Python interface to Scikit-learn for Neuroimaging. Easy-to-use/install. Good viz.
- **Pymvpa** Python tool for ML. Advanced stuff (Pipelines, Hyperalignment).