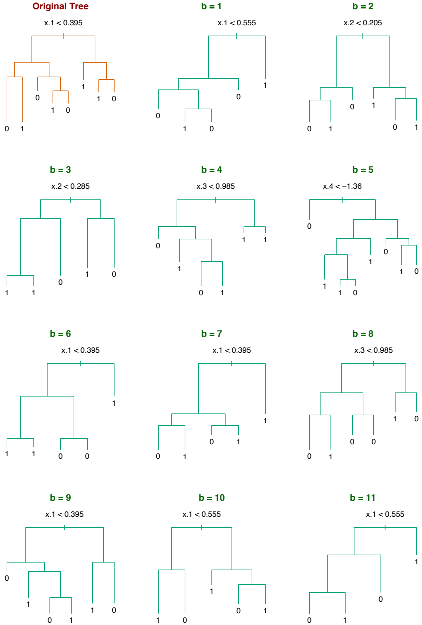
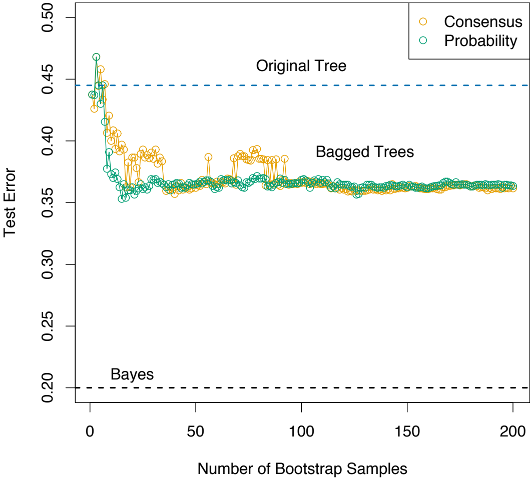
EXAMPLE: BAGGING TREES



- Two classes, each with Gaussian distribution in $\mathbb{R}^5$.
- Note the variance between bootstrapped trees.

EXAMPLE: BAGGING TREES



- "Original tree" = single tree trained on original data.
- The orange dots correspond to the bagging classifier.

Bagging vs. Boosting

- Bagging works particularly well for trees, since trees have high variance.
- Boosting typically outperforms bagging with trees.
- The main culprit is usually dependence: Boosting is better at reducing correlation between the trees than bagging is.

Random Forests

Modification of bagging with trees designed to further reduce correlation.

- Tree training optimizes each split over all dimensions.
- Random forests choose a different subset of dimensions *at each split*.
- Optimal split is chosen within the subset.
- The subset is chosen at random out of all dimensions $\{1, \ldots, d\}$.

Training

Input parameter: m (positive integer with $m < d$)

For $b = 1, \ldots, B$:

1. Draw a bootstrap sample $\mathcal{B}_b$ of size n from training data.
2. Train a tree classifier f_b on $\mathcal{B}_b$, where each split is computed as follows:
    - Select m axes in $\mathbb{R}_d$ at random.
    - Find the best split (j^*, t^*) on this subset of dimensions.
    - Split current node along axis j^* at t^*.

Classification

Exactly as for bagging: Classify by majority vote among the B trees. More precisely:

- Compute $f_{\text{avg}}(\mathbf{x}) := (p_1(\mathbf{x}), \ldots, p_k(\mathbf{x})) := \frac{1}{B} \sum_{b=1}^{B} f_b(\mathbf{x})$
- The Random Forest classification rule is

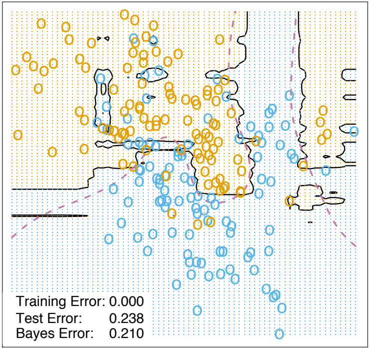
$$f_{\text{Bagging}}(\mathbf{x}) := \arg\max_k \{p_1(\mathbf{x}), \ldots, p_k(\mathbf{x})\}$$

Remarks

- Recommended value for m is $m = \lfloor \sqrt{d} \rfloor$ or smaller.
- RF typically achieve similar results as boosting. Implemented in most packages, often as standard classifier.

Example: Synthetic Data

- This is the RF classification boundary on the synthetic data we have already seen a few times.
- Note the bias towards axis-parallel alignment.



SUMMARY: CLASSIFICATION

SUMMARY

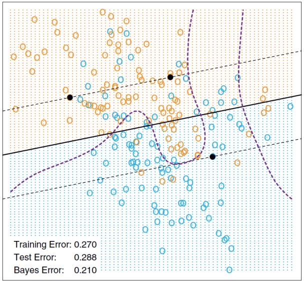
Approaches we have discussed

- Linear classifiers
  - Perceptron, SVM
  - Nonlinear versions using kernels
- Trees (depth 1: linear and axis-parallel, depth ≥ 2: non-linear)
- Ensemble methods

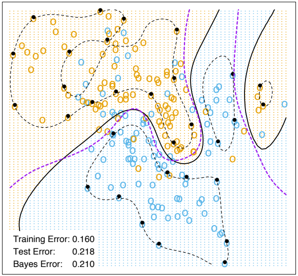
What should we use?

- RBF SVMs, AdaBoost and Random Forests perform well on many problems.
- All have strengths and weaknesses. E.g.:
  - High dimension, limited data: SVM may have the edge.
  - Many dimensions, but we believe only a few are important: AdaBoost with stumps.
- In general: Feature extraction (what do we measure?) is crucial.
- Consider combination of different methods by voting.

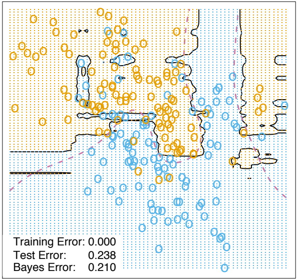
EVERY METHOD HAS ITS IDIOSYNCRASIES



Linear SVM



RBF SVM



Random forest

REGRESSION

REGRESSION: PROBLEM DEFINITION

Data

- Measurements: $\mathbf{x} \in \mathbb{R}^d$ (also: independent variable, covariate)
- Labels: $y \in \mathbb{R}$ (also: dependent variable, response)

Task

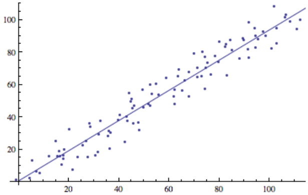
Find a predictor $f : \mathbb{R}^d \to \mathbb{R}$ such that (approximately) $f(x) = y$ for data (x, y). The predictor is called a **regression function**.

Definition: Linear regression

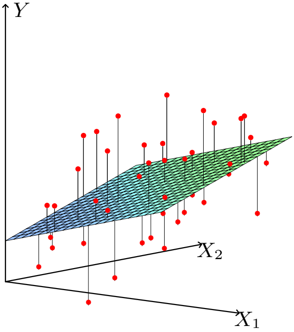
A regression method is called **linear** if the predictor f is a linear function, i.e. a line if $d = 1$ (more generally, an affine hyperplane).

Linear Regression

$$\mathbf{x} \in \mathbb{R}^d \quad \text{and} \quad y \in \mathbb{R}$$



$d = 1$



$d = 2$

Implications of linearity

A linear function $f : \mathbb{R}^d \to \mathbb{R}$ is always of the form

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j \qquad \text{for } \beta_0, \beta_1, \dots, \beta_d \in \mathbb{R} \,,$$

where x_j is the jth entry of $\mathbf{x}$. Recall representation of hyperplanes in classification!

Consequence

Finding f boils down to finding $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$.

Relation to classification

- Classification is a regression problem with $\{1, \dots, K\}$ substituted for $\mathbb{R}$.
- Don't get confused—the role of the hyperplane (for, say, $d = 2$) is different:
  - Regression: Graph of regression function is hyperplane in $\mathbb{R}^{d+1}$.
  - Classification: Regression function is piece-wise constant. The classifier hyperplane lives in $\mathbb{R}^d$ and marks where the regression function jumps.

Squared-error loss

We use the **squared-error loss function**

$$L^{\text{se}}(y, f(x)) := \|y - f(x)\|_2^2 \; .$$

Regression methods that determine f by minimizing L^{se} are called **least-squares regression** methods.

Least-squares linear regression

For training data $(\tilde{\mathbf{x}}_1, \tilde{y}_1), \ldots, (\tilde{\mathbf{x}}_n, \tilde{y}_n)$, we have to find the parameter vector $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$ which solves

$$\hat{\boldsymbol{\beta}} := \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} L^{\text{se}}(\tilde{y}_i, f(\tilde{\mathbf{x}}_i\,; \boldsymbol{\beta}))$$

where

$$f(\mathbf{x}\,; \boldsymbol{\beta}) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j = \langle \boldsymbol{\beta}, (1, \mathbf{x}) \rangle \; .$$

MATRIX FORMULATION

Data matrix

Since $f(\mathbf{x}\,;\boldsymbol{\beta}) = \langle \boldsymbol{\beta}, (1, \mathbf{x}) \rangle$, we write the data as a matrix:

$$\tilde{\mathbf{X}} := \begin{array}{cc} \begin{pmatrix} 1 & (\tilde{\mathbf{x}}_1)_1 & \dots & (\tilde{\mathbf{x}}_1)_j & \dots & (\tilde{\mathbf{x}}_1)_d \\ \vdots & & & \vdots & & \vdots \\ 1 & (\tilde{\mathbf{x}}_i)_1 & \dots & (\tilde{\mathbf{x}}_i)_j & \dots & (\tilde{\mathbf{x}}_i)_d \\ \vdots & & & \vdots & & \vdots \\ 1 & (\tilde{\mathbf{x}}_n)_1 & \dots & (\tilde{\mathbf{x}}_n)_j & \dots & (\tilde{\mathbf{x}}_n)_d \end{pmatrix} & \begin{array}{c} \\ \\ \longleftarrow \tilde{\mathbf{x}}_i \\ \\ \\ \end{array} \end{array}$$

We write $\tilde{\mathbf{X}}_j^{\text{col}}$ for the column vectors with $\tilde{\mathbf{X}}_0^{\text{col}} = (1, \dots, 1)$ and $j = 1, \dots, d$.

$$\tilde{\mathbf{X}}\boldsymbol{\beta} = \begin{pmatrix} f(\tilde{\mathbf{x}}_1\,;\boldsymbol{\beta}) \\ \vdots \\ f(\tilde{\mathbf{x}}_n\,;\boldsymbol{\beta}) \end{pmatrix}$$

Least-squares linear regression: Matrix form

We have to minimize

$$\sum_{i=1}^{n} L^{\text{se}}(\tilde{y}_i, f(\tilde{\mathbf{x}}_i;\boldsymbol{\beta})) = \sum_{i=1}^{n} (\tilde{y}_i - f(\tilde{\mathbf{x}}_i;\boldsymbol{\beta}))^2 = \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}\|_2^2$$

The solution to the linear regression problem is now $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}\|^2$.

Solving the minimization problem

Recall:

- We have to solve for a zero derivative, $\frac{\partial L^{\text{se}}}{\partial \boldsymbol{\beta}}(\hat{\boldsymbol{\beta}}) = 0$.
- That means that $\hat{\boldsymbol{\beta}}$ is an extremum.
- To ensure that the extremum is a minimum, we have to ensure the second derivative $\frac{\partial^2 L^{\text{se}}}{\partial \boldsymbol{\beta}^2}(\hat{\boldsymbol{\beta}})$ is positive. For matrices: Positive definite.

LEAST-SQUARES SOLUTION

Solution

$$\frac{\partial L^{\text{se}}}{\partial \boldsymbol{\beta}}(\hat{\boldsymbol{\beta}}) = -2\tilde{\mathbf{X}}^t(\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta})$$

Equating to zero gives the **least-squares solution**:

$$\hat{\boldsymbol{\beta}} = (\tilde{\mathbf{X}}^t\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^t y$$

(Recall: The transpose $\mathbf{X}^t$ is the matrix with $(\mathbf{X}^t)_{ij} := \mathbf{X}_{ji}$.)

Second derivative

$$\frac{\partial^2 L^{\text{se}}}{\partial \boldsymbol{\beta}^2}(\hat{\boldsymbol{\beta}}) = 2\tilde{\mathbf{X}}^t\tilde{\mathbf{X}}$$

- $\tilde{\mathbf{X}}^t\tilde{\mathbf{X}}$ is always positive semi-definite. If it is also invertible, it is positive definite.
- In other words: If $\tilde{\mathbf{X}}^t\tilde{\mathbf{X}}$ is invertible (which we also need to compute $\hat{\boldsymbol{\beta}}$), then $\hat{\boldsymbol{\beta}}$ is the unique global minimum of the squared-error loss.

TOOLS: LINEAR ALGEBRA BASICS

Images of Linear Mappings (1)

Linear mapping

A matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ defines a linear mapping $f_{\mathbf{X}} : \mathbb{R}^m \to \mathbb{R}^n$.

Image

Recall: The **image** of a mapping f is the set of all possible function values, here

$$\text{image}(f_{\mathbf{X}}) := \{\mathbf{y} \in \mathbb{R}^n \,|\, \mathbf{Xz} = \mathbf{y} \text{ for some } \mathbf{z} \in \mathbb{R}^m\}$$

Image of a linear mapping

- The image of a linear mapping $\mathbb{R}^m \to \mathbb{R}^n$ is a linear subspace of $\mathbb{R}^n$.
- The columns of $\mathbf{X}$ form a basis of the image space:

$$\text{image}(\tilde{\mathbf{X}}) = \text{span}\{\mathbf{X}_1^{\text{col}}, \ldots, \mathbf{X}_m^{\text{col}}\}$$

- This is one of most useful things to remember about matrices, so, again:

The columns span the image.

IMAGES OF LINEAR MAPPINGS (2)

Dimension of the image space

Clearly: The number of linearly independent column vectors. This number is called the **column rank** of $\mathbf{X}$.

Invertible mappings

Recall: A mapping f is invertible if it is one-to-one, i.e. for each function value $\tilde{\mathbf{y}}$ there is exactly one input value with $f(\mathbf{z}) = \tilde{\mathbf{y}}$.

Invertible matrices

The matrix $\tilde{\mathbf{X}}$ is called invertible if $f_{\mathbf{X}}$ is invertible.

- Only square matrices can be invertible.
- For a *linear* mapping: If $\tilde{\mathbf{X}}$ is a square matrix $f_{\mathbf{X}}$ is invertible if the image has the same dimension as the input space.
- Even if $\tilde{\mathbf{X}} \in \mathbb{R}^{n \times m}$, the matrix $\tilde{\mathbf{X}}^t\tilde{\mathbf{X}}$ is in $\mathbb{R}^{m \times m}$ (a square matrix).
- So: $\tilde{\mathbf{X}}^t\tilde{\mathbf{X}}$ is invertible if $\tilde{\mathbf{X}}$ has full column rank.

Symmetric and Orthogonal Matrices

Recall: Transpose

The transpose A^{T} of a matrix $A \in \mathbb{R}^m$ is the matrix with entries

$$(A^{\mathrm{T}})_{ij} := A_{ji}$$

Orthogonal matrices

A matrix $O \in \mathbb{R}^{m \times m}$ is called **orthogonal**

$$O^{-1} = O^T$$

Orthogonal matrices describe two types of operations:

1. Rotations of the coordinate system.
2. Permutations of the coordinate axes.

Symmetric matrices

A matrix $A \in \mathbb{R}^{m \times m}$ is called **symmetric**

$$A = A^{\mathrm{T}}$$

Note: Symmetric and orthogonal matrices are very different objects. Only the identity is both.

ORTHONORMAL BASES

Recall: ONB

A basis $\{v_1, \dots, v_m\}$ of $\mathbb{R}^m$ is called an **orthonormal basis** if

$$\langle v_i, v_j \rangle = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

In other words, the v_i are pairwise orthogonal and each of length 1.

Orthogonal matrices

A matrix is orthogonal precisely if its rows form an ONB. Any two ONBs can be transformed into each other by an orthogonal matrix.

Representation of a vector

Suppose $\mathcal{E} = \{e_1, \ldots, e_d\}$ is a basis of a vector space. Then a vector x is represented as

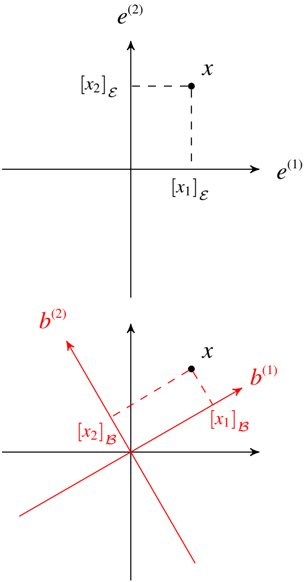
$$x = \sum_{j=1}^{d} [x_j]_{\mathcal{E}} e^{(j)}$$

$[x_j]_{\mathcal{E}} \in \mathbb{R}$ are the coordinates of x w.r.t. $\mathcal{E}$.

Other bases

If $\mathcal{B} = \{b_1, \ldots, b_d\}$ is another basis, x can be represented alternatively as

$$x = \sum_{j=1}^{d} [x_j]_{\mathcal{B}} b^{(j)}$$



Change-of-basis matrix

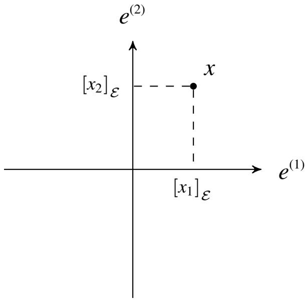
The matrix

$$M := \left([e^{(1)}]_{\mathcal{B}}, \dots, [e^{(d)}]_{\mathcal{B}} \right)$$

transforms between the bases, i.e.

$$M[x]_{\mathcal{E}} = [x]_{\mathcal{B}} \ .$$

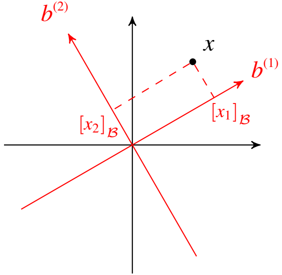
If both $\mathcal{E}$ and $\mathcal{B}$ are ONBs, M is orthogonal.



Representation of matrices

The matrix representing a linear mapping $A : \mathbb{R}^d \to \mathbb{R}^d$ in the basis $\mathcal{E}$ is computed as

$$[A]_{\mathcal{E}} := \left([A(e^{(1)})]_{\mathcal{E}}, \dots, [A(e^{(d)})]_{\mathcal{E}} \right)$$



Transforming matrices

The matrix representing a linear mapping also changes when we change bases:

$$[A]_{\mathcal{B}} = M[A]_{\mathcal{E}} M^{-1} \ .$$

Applied to a vector x, this means:

apply A in representation $\mathcal{E}$

transform x from $\mathcal{B}$ to $\mathcal{E}$

$$[A]_{\mathcal{B}}[x]_{\mathcal{B}} = M[A]_{\mathcal{E}} M^{-1}[x]_{\mathcal{B}} \ .$$

transform x back to $\mathcal{B}$

Transforming between ONBs

If $\mathcal{V} = \{v_1, \ldots, v_m\}$ and $\mathcal{W} = \{w_1, \ldots, w_m\}$ are any two ONBs, there is an orthogonal matrix O such that

$$[A]_{\mathcal{V}} = O[A]_{\mathcal{W}} O^{-1}$$

for any linear mapping A.