Summary Advanced Machine Learning

Representations, Measurements, Data types

- Problem of overfitting
  - Minimize empirical classification error
    ↳ ≈ true classification error if n is big enough
  - Maximize estimated empirical generalization performance by cross validation
    ↳ ≈ generalization
- Training error: $\hat{R}(\hat{f}, z^{Train}) = \frac{1}{2} \sum_{i=1}^{n} Q(y_i, \hat{f}(x_i))$ (Loss function; $\hat{f}$ ↑ model)
- Test error: only with validation data
- Expected risk: $R(f) = E_x[R(f, x)]$

↳ Test error estimates expected risk but it has a slight tendency to be optimistic

- Matrix Cook Book
  - $(A \cdot B \cdot C)^{-1} = C^{-1} \cdot B^{-1} \cdot A^{-1}$
  - $(A + B)^{-1} = A^{-1} + B^{-1}$
  - $(A^T)^{-1} = (A^{-1})^T$
  - $A^{-1}A = I$

  ⊢
  - $(A + B)^T = A^T + B^T$
  - $(A \cdot B \cdot C)^T = C^T \cdot B^T \cdot A^T$

  ⊢
  - $\|x\|_1 = \sum_i |x_i|$ (= sum of absolute vectors)
  - $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \dots}$
  - $\|x\|_2^2 = x^T x$

  ⊢
  - $\frac{\partial}{\partial x} x^T a = \frac{\partial}{\partial x} a^T x = a$
  - $\frac{\partial}{\partial x} a^T x b = a b^T$
  - $\frac{\partial}{\partial x} a^T x^T b = \frac{\partial}{\partial x} b^T x a = b a^T$

- Least squares

$$\cdot RSS(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - x_i^T \cdot \beta)^2$$

$$= (y - X\beta)^T \cdot (y - X\beta)$$

$$= (y^T - \beta^T X^T) \cdot (y - X\beta)$$

$$= y^T y - y^T X\beta - \beta^T X^T y + \beta^T X^T X\beta$$

$$= y^T y - 2y^T X\beta + \beta^T X^T X\beta$$

$$\cdot \frac{\partial RSS(\beta)}{\partial \beta} = -2X^T y + 2X^T X\beta \stackrel{!}{=} 0 \quad / + 2X^T y \; / :2$$

$$X^T X\beta = X^T y \quad / (X^T X)^{-1}$$

$$\hat{\beta} = (X^T X)^{-1} \cdot X^T y$$

- K-Fold cross validation algorithm

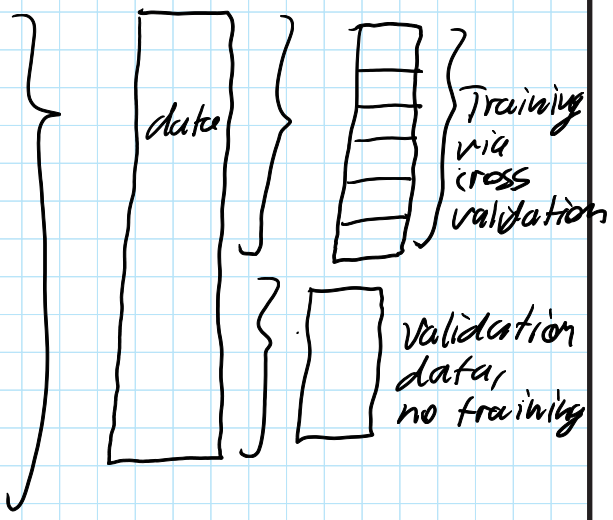
Algorithm 1 K-Fold Cross Validation

initialize: Split data $\mathcal{X}$ into K equally sized subsets:

$$\mathcal{X} = \bigcup_{i=1}^{K} \mathcal{X}_i \qquad \text{s.t.} \quad \forall \mu \neq \nu \quad \mathcal{X}_\mu \cap \mathcal{X}_\nu = 0$$

for $i = 1, \ldots, K$ **do**
  Train model on data $\mathcal{X}_{-i} = \mathcal{X} \backslash \mathcal{X}_i$
  Estimate its prediction error $\mathcal{E}_i$ on $\mathcal{X}_i$
end for
Combine all estimates:

$$\mathcal{E}_{CV} = \frac{1}{K} \sum_{i=1}^{K} \mathcal{E}_i$$



↳ for tuning hyperparameters e.g. λ

- Discriminative Modeling
  - $P(y|x)$
  - Can be derived by generative modeling using Bayes' rule
  - Labeled learners, can't make use of unlabeled data

- Generative Modeling
  - $P(y, x)$
  - Can detect outliers because we know the probability x is chosen
  - Use unlabeled data

Density Estimation

Bayesianism

- $P(\text{model} \mid \text{data}) = \frac{P(\text{data} \mid \text{model}) \cdot P(\text{model})}{P(\text{data})}$ (likelihood, prior, evidence; can be left out)
  - posterior → given the data, find best model
- We can define a prior on the model (e.g. mean) to give it robustness: $P(\theta) = \mathcal{N}(42, 1)$
- The posterior's mean is a weighted sum of the sample mean and the prior's mean
- The prior acts as a "regularization term"
- In Bayesian linear regression we are doing least-squares-regression with a regularization term
- If prior assigns probability 0 to value, posterior of
- Sequential Bayesian inference that value is 0, too!
  - We can update our probability distributions with additional data
  - The posterior-distribution after observing n-1 data points can be viewed as the prior distribution, where the likelihood function is associated with the new data point x_n

Frequentism

- We don't have a prior
- Maximum likelihood method
  - ↳ Select the parameters $\hat{\theta}$ which maximizes the likelihood: $\hat{\theta} \in \arg\max P(x \mid \theta)$
- Provides a single point when estimating parameters
- $P(\text{Data} \mid \text{model}) = \prod P(D_i \mid \text{model})$
  - ↳ Maximum Likelihood: Tune model parameters to get highest probability

Good estimators

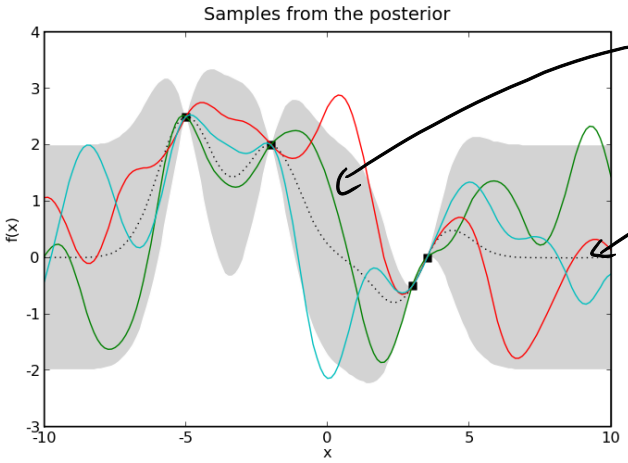
- Unbiased: $E[\hat{\theta}] - \theta_0 \hat{=} 0$
- Efficient: $E[(\hat{\theta} - \theta_0)^2] \hat{=} \frac{1}{I_n(\theta_0)}$
- Consistent: when $n \to \infty$

Regression

- Maximum Likelihood $\hat{=}$ Linear regression
  - ↳ $p(y|X, \beta, \sigma) = \mathcal{N}(y|X^T\beta, \sigma^2)$
    $y \sim \mathcal{N}(X^T\beta, \sigma^2) \Rightarrow y \hat{=}$ mean
- Maximum A Posteriori $\hat{=}$ Lasso
- An estimator is unbiased, if the difference between estimator's expected value and the true value is 0: $E[x] = x$ (=unbiased)
- Bias/Variance tradeoff
  - Trade bias for variance decrease
  - Bias $\hat{=}$ Unteranpassung
  - Variance $\hat{=}$ Überanpassung
- Bayesian linear Regression
  - $y \sim \mathcal{N}(\underbrace{X\beta}_{\text{Mean}}, \underbrace{\sigma^2 I}_{\text{variance}})$
  - By choosing a suitable prior we can achieve
    - Ridge regression: $\beta \sim \mathcal{N}(0, \sigma^2/\lambda I)$
    - Lasso: $p(\beta_i) = \frac{\lambda}{4\sigma^2} \cdot \exp(-|\beta| \frac{\lambda}{2\sigma^2}$
    - Standard Linear regression: Non-informative priors
      ↳ e.g. uniform

Gaussian Processes

- Probabilistic method that gives a confidence for the predicted value



The gray area are all functions

Mean of all functions

- Infinitely many possible functions for data-set
- We want to assign each function a probability
- The mean of all functions: Best fitting function
- Bayesian Linear Regression: · All about the weights
  · The mean minimizes SSR
- Gaussian Processes: · (x, y) are now gaussians
  · The median is the best y out of all functions

Numerical Estimation Techniques

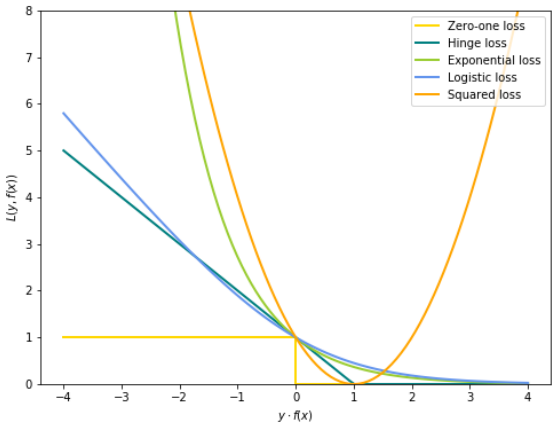
- Expected loss: $R(f) = E_{x,y}[(y - f(x))^2]$
- Empirical loss: $\hat{R}(f) = \frac{1}{n} \sum (y_i - f(x_i))^2$
- **Bootstrapping**
  - Resampling with replacement
  1. Create Bootstrap sample sets by drawing with replacement from training set
  2. Train model for each sample set
  3. Take the mean over all trained models for e.g. risk
  - Data source F_x is approximated by empirical data source $\hat{F}_x$
  - Naive Bootstrap error estimation
    - $\hat{R}^* = \frac{1}{B} \cdot \frac{1}{n} \sum \sum l(y_i, \hat{f}^{*b}(x_i))$
    - Problem: Overlap of training / testing data
      ↳ Too optimistic
  - Leave-one-out Bootstrap error estimation
    - $\hat{R}^* = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} l(y_i, f^{*b}(x_i))$
    - Loops through each (x_i, y_i) pair
    - If pair is in one bootstrap sample set, we exclude it from loss function, since we trained it on that sample
    - C^{-i}: all bootstrap sample sets which don't contain that sample
      ↳ No overlap between Train/Test
  - Based on law of large numbers, if you sample over and over again, your data should approximate the true distribution

- Jackknife (special case of Bootstrapping, no replacement)
  - Goal: estimation of the bias of an estimator
  - Aus der ursprünglichen Stichprobe wird jeweils ein Wert weggelassen und ein Schätzer dafür berechnet
  - Leave-one-out Estimator: $\hat{S}_{n-1}^{(-i)}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$
  - $\hat{\theta}_{JK} = \frac{1}{n} \sum_{i=1}^{n} \hat{\theta}_{(i)}$

    $\hat{\theta}_{(i)} = \frac{1}{n-1} \sum_{\substack{j=1 \\ j \neq i}}^{n} x_j$
  - $\hat{\theta}$: estimator over all n observations
  - $\widehat{bias_{(\theta)}} = (n-1)(\hat{\theta}_{JK} - \hat{\theta})$
  - $\hat{\theta}_{jack} = \hat{\theta} - \widehat{bias_{(\theta)}}$
  - Bias corrected estimators can have larger variance!

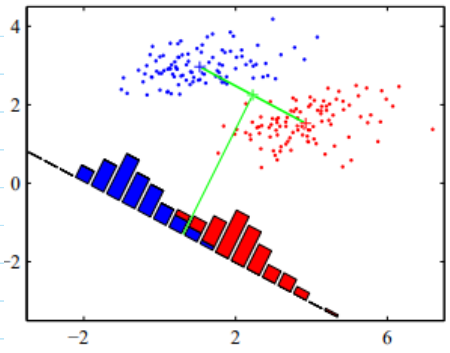
Linear discriminant function

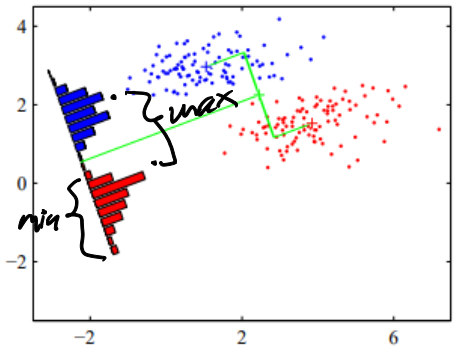
- Used for classifying dichotomies
- D: doubt class, need more training
- O: outlier class, not inside any class
- $\mathbb{I}$: indicator function: $1 \Leftrightarrow \hat{C}(x_i) \neq y_i$
- Weighted mistakes:
  - Used for medical diagnosis
  - Weighted losses or weighted classification costs
- 0-1 loss is non-differentiable (non-convex)
  ↳ use surrogate loss function instead e.g. hinge loss



- Gradient descend: uses only first order derivative
- Newton's Method: Uses also second order derivatives
  - We are approximating the function at x_t and setting x_{t+1} to the minimum of that function
    ↳ second order approximation
- We can look at both methods as first and second order function approximation methods, where the second order is much faster
  ↳ Taylor expansion

- (single layer) perceptron algorithm
  1. We are starting with random weights for classification hyperplane
  2. We select all wrongly classified training points and update weights
     ↳ $w \leftarrow w + \sum_{x \in X^{MC}} \eta(k)\tilde{x}$
     ↑ Learning rate at step k
  - Instead of batch we could use single sample where we only use only one misclassified training sample
  - Only converges, if data is linear seperable
- Fisher's linear discriminant analysis
  - Classification by dimensionality reduction
  - solves classification problem with k classes in an k-1 dimensional subspace
  - Dichotomy is solved by projecting on a line



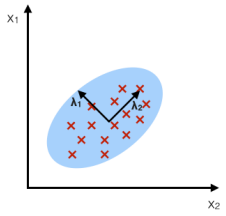


  - Variance of class is minimized
  - Distance between means of classes maximized

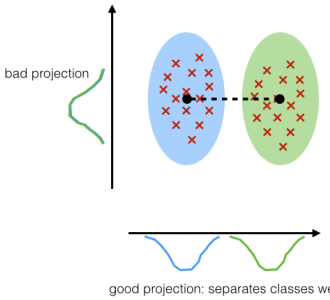
- Linear discriminant analysis: · Max. Intercluster Variance
  · Min. Intracluster Variance
  · Good, if classes are Gaussian distributed
- Perceptron: · More naive approach
  · Infinetily many hyperplanes
  · e.g. SVM

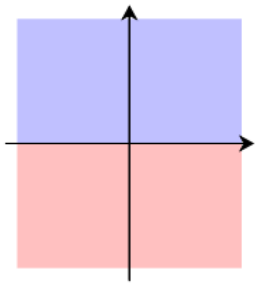
↳ Both support kernels for non-linear boundaries

PCA:
component axes that maximize the variance

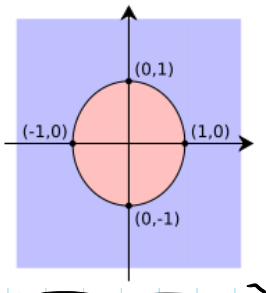


LDA:
maximizing the component axes for class-separation

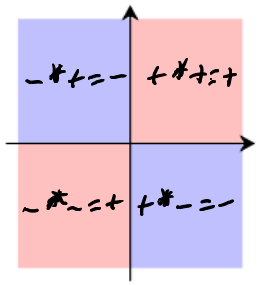




$w_1 = 0$
$w_2 = 1$

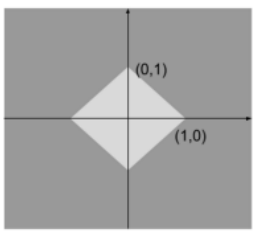


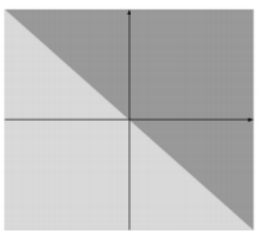
$\phi(x,y) = x^2 + y^2$
$w = [-1, 1]$
$\hat{x} = [1, \phi(x,y)]$

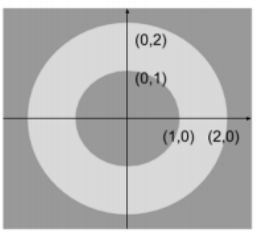


$\phi(x,y) = x \cdot y$
$w_1 = 1$
$w_2 = 1$

$w_1 \cdot x + w_2 \cdot x \gtrless 0$







$\phi(x,y) = |x| + |y|$
$w = [-1, 1]$
$\hat{x} = [1, \phi(x,y)]$

$w_1 = 1$
$w_2 = 1$

Support Vector Machines

- Perceptron with Margin and Kernel
- Lagrange multipliers
  - Optimization with sideconditions
  - Only equalities as sideconditions
  1. $f(x) + \lambda_1 \cdot g_1(x) + \lambda_2 \cdot g_2(x)$
  2. Solve for $\frac{\partial}{\partial x} = 0$ and $\frac{\partial}{\partial \lambda} = 0$
  3. Insert λ value into $\frac{\partial}{\partial x} = 0$ to get x^*
- KKT conditions
  - Extension of Lagrange Multipliers
  - Adds the inequality as sidecondition case

- The Lagrangian is:

$$L(x_1, y, \mu_1, \mu_2) = -(x-2)^2 - 2(y-1)^2 + \mu_1(3-x-4y) + \mu_2(0+x-y)$$

- This gives the following KKT conditions

$$\frac{\partial L}{\partial x} = -2(x-2) - \mu_1 + \mu_2 = 0$$

$$\frac{\partial L}{\partial y} = -4(y-1) - 4\mu_1 - \mu_2 = 0$$

$$\mu_1(3-x-4y) = 0$$

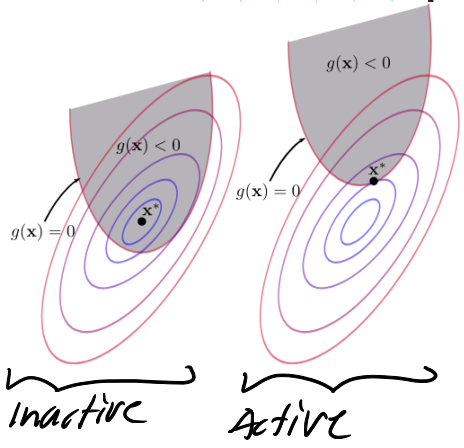
$$\mu_2(x-y) = 0$$

$$\mu_1, \mu_2 \geq 0$$

1. $\mu_1 = \mu_2 = 0 \rightarrow x = 2, y = 1$ ✓ => Not optimal
2. $\mu_1 = 0, x - y = 0 \rightarrow x = \frac{4}{3}, \mu_2 = -\frac{4}{3}$ ↯ $\mu_2 < 0$
3. $3 - x - 4y = 0, \mu_2 = 0 \rightarrow x = \frac{5}{3}, y = \frac{1}{3}, \mu_1 = \frac{2}{3}$ ✓ => optimal
4. $3 - x - 4y = 0, x - y = 0 \rightarrow x = \frac{3}{5}, y = \frac{3}{5}, \mu_1 = \frac{22}{25}, \mu_2 = -\frac{48}{25}$ ↯ $\mu_2 < 0$

Optimal solution is therefore $x^* = \frac{5}{3}, y^* = \frac{1}{3}, f(x^*, y^*) = -\frac{4}{9}$

- Inactive constraints: $g(x^*) > 0$
  ↳ function $g(x)$ plays no role
- Active constraints: $g(x^*) = 0$
  ↳ function plays a role



-Duality Theorem

$$\mathcal{P}: \quad \begin{array}{lll} \min & f(\mathbf{w}), & \mathbf{w} \in \mathbb{R}^d \\ \text{s.t.} & g_i(\mathbf{w}) = 0, & i \leq m \\ & h_j(\mathbf{w}) \leq 0, & j \leq n \end{array}$$

$$\tilde{\mathcal{P}}: \quad \begin{array}{lll} \max & \theta(\boldsymbol{\lambda}, \boldsymbol{\alpha}), & \boldsymbol{\lambda}, \boldsymbol{\alpha} \in \mathbb{R}^{m+n} \\ \text{s.t.} & \alpha_i \geq 0, & i \leq n \end{array}$$

- Primal
- Dual
- Weak duality theorem: $f(w^*) \geq \theta(\lambda^*, \alpha^*)$
- Strong duality theorem: $f(w^*) = \theta(\lambda^*, \alpha^*)$
  - ↳ Assuming feasable w where $h_j(w) < 0$

- Hard Margin SVM

- We want to maximize margin: $2\frac{m}{\|w\|}$
  - Geometric margin problem: maximize m for $\|w\| = 1$
  - Functional margin problem: minimize $\|w\|$ for $m = 1$

$$\begin{array}{lll} \text{minimize} & \mathcal{T}(\mathbf{w}) & = \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} \\ \text{subject to} & \forall i: \; z_i(\mathbf{w}^\mathsf{T}\mathbf{y}_i + w_0) & \geq 1 \end{array}$$

$\frac{\partial L}{\partial w}$ and $\frac{\partial L}{\partial w_0}$

$$L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} - \sum_{i \leq n} \alpha_i \left[z_i(\mathbf{w}^\mathsf{T}\mathbf{y}_i + w_0) - 1 \right]$$

$$= \frac{1}{2} \sum_{i \leq n} \sum_{j \leq n} \alpha_i \alpha_j z_i z_j \mathbf{y}_i^\mathsf{T}\mathbf{y}_j - \sum_{i \leq n} \sum_{j \leq n} \alpha_i \alpha_j z_i z_j \mathbf{y}_i^\mathsf{T}\mathbf{y}_j + \sum_{i \leq n} \alpha_i$$

$$= \sum_{i \leq n} \alpha_i - \frac{1}{2} \sum_{i \leq n} \sum_{j \leq n} \alpha_i \alpha_j z_i z_j \mathbf{y}_i^\mathsf{T}\mathbf{y}_j \qquad \text{(note the scalar product!)}$$

- Dual hard margin SVM: $\begin{array}{ll} \text{maximize} & W(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n z_i z_j \alpha_i \alpha_j \mathbf{y}_i^\mathsf{T}\mathbf{y}_j \\ \text{subject to} & \forall i: \; \alpha_i \geq 0 \quad \wedge \quad \sum_{i=1}^n z_i \alpha_i = 0 \end{array}$

  ↳ $w = \sum_{i=1}^{m} \alpha_i \cdot y_i \cdot x_i$

- Faster to solve, because α correspond to data points
  - ↳ High dimension, few data points
  - w, α
- $\alpha_i^* \geq 0$ Support vector point
- $\alpha_i^* = 0$ Non support vector point

- Soft Margin SVM

- Introduce slack variable $\xi_i \geq 0$

$$\begin{array}{rrcl} \text{minimize} & \mathcal{T}(\mathbf{w}, \boldsymbol{\xi}) & = & \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} + C\sum_{i=1}^n \xi_i \\ \text{subject to } \forall i: & z_i(\mathbf{w}^\mathsf{T}\mathbf{y}_i + w_0) & \geq & 1 - \xi_i \\ & \xi_i & \geq & 0 \end{array}$$

C ← constraint violation vs. margin maximization ⇓ cross validation

$$\begin{aligned} L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \tfrac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j z_i z_j \mathbf{y}_i^\mathsf{T}\mathbf{y}_j + C\sum_{i=1}^n \xi_i \\ &\quad - \sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j z_i z_j \mathbf{y}_i^\mathsf{T}\mathbf{y}_j \\ &\quad + \sum_{i=1}^n \alpha_i(1-\xi_i) - \sum_{i=1}^n \beta_i\xi_i \\ &= \sum_{i=1}^n \alpha_i - \tfrac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j z_i z_j \mathbf{y}_i^\mathsf{T}\mathbf{y}_j \\ &\quad + \sum_{i=1}^n \underbrace{(C - \alpha_i - \beta_i)}_{=\frac{\partial L}{\partial \xi_i}=0}\xi_i \\ &= \sum_{i=1}^n \alpha_i - \tfrac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j z_i z_j \mathbf{y}_i^\mathsf{T}\mathbf{y}_j \end{aligned}$$

- Dual soft margin SVM

$$\begin{array}{ll} \text{maximize} & W(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n z_i z_j \alpha_i\alpha_j \mathbf{y}_i^\mathsf{T}\mathbf{y}_j \\ \text{subject to} & \sum_{j=1}^n z_j\alpha_j = 0 \ \wedge \ \forall i \ C \geq \alpha_i \geq 0 \end{array}$$

Two tricks to simplify notation:

1. Subsume w_0: augment feature vector $\mathbf{x} \in \mathbb{R}^D$ into $\tilde{\mathbf{x}} := (\mathbf{x}, 1) \in \mathbb{R}^{D+1}$

$$w_0 + \sum_{d=1}^{D} w_d x_d = \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}$$

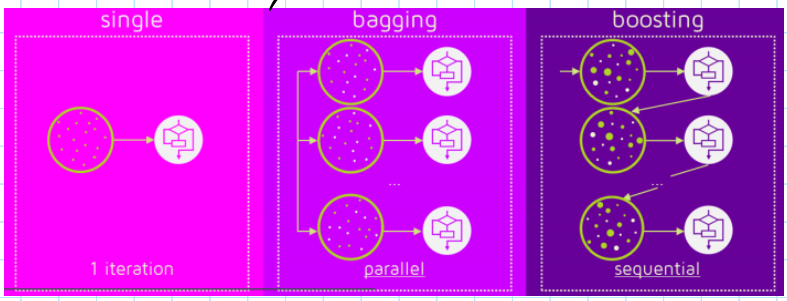
with $\tilde{\mathbf{w}} = (\mathbf{w}, w_0)$.

2. Replace two sided test with one inequality: $\mathbf{x}_{(n)}$ correctly classified if

$$y_{(n)} f(\mathbf{x}_{(n)}) > 0.$$

$c : \Omega \times \mathcal{X} \to \{-1, 1\}$

Ensemble Methods



- Boosting plays with weights, Bagging doesn't

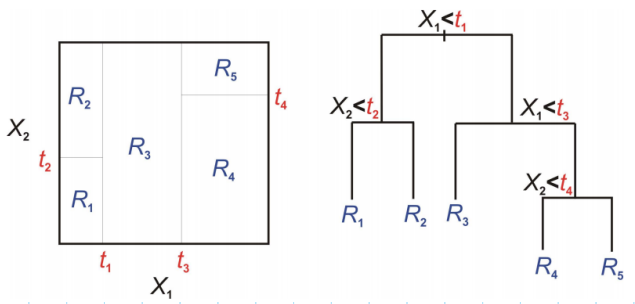
- **Bagging**:
  - Bootstrap aggregation
  - Combines several samples which have been trained on random subsamples with replacement by weak learners

INPUT: data $(x_1, y_1), \ldots, (x_n, y_n)$
OUTPUT: classifier with majority vote of $\{c_1, c_2, \ldots, c_B\}$

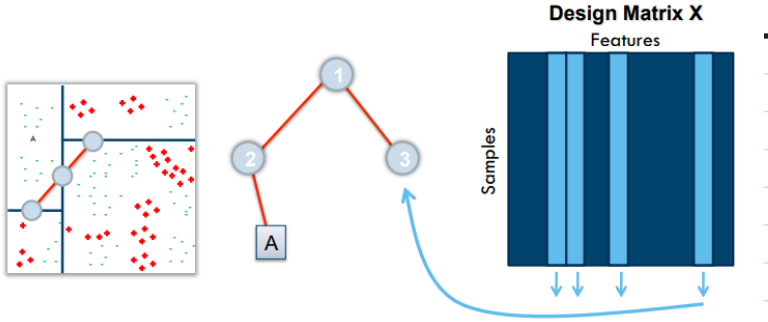
1: **for** $b = 1$ to B **do** ← weak learners
2: $\mathcal{Z}^{\star b} =$ b-th bootstrap sample from $\mathcal{Z}$
3: Construct classifier c_b based on $\mathcal{Z}^{\star b}$
4: **end for**
5: **return** ensemble classifier $\hat{c}_B(x) = \text{sgn}\left(\sum_{i=1}^{B} \alpha_i c_i(x)\right)$ ($\alpha_i = 1$)

- Two methods proposed for dealing with imbalanced training data
  1. Class weights
     - Give greater weight to minority class samples
  2. balanced bootstrapping (balanced random forests)
     - 2.1 Take bootstrap sample of minority class
     - 2.2 Sample of same size from majority class

- **Decision Trees**

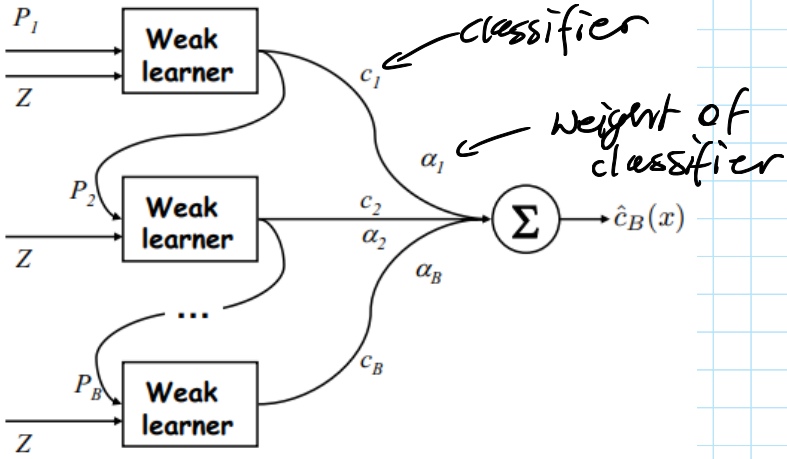


1: **for** $b = 1$ to B **do**
2: draw a bootstrap sample $\mathcal{Z}^\star = \{(X_1^\star, Y_1^\star), \ldots, (X_n^\star, Y_n^\star)\}$ of size n from the training data $\mathcal{Z}$;
3: **repeat**
4: select m variables at random from p variables $(m \approx \sqrt{p})$; (=feature selection)
5: pick the best variable/split-point among the m selected variables;
6: split the node into two daughter nodes;
7: **until** node size n_{min} is reached
8: **end for**
9: **return** Output the ensemble of trees $\{\hat{c}_b(x)\}_{b=1}^{B}$



- Random forest: a collection of decision trees trained on bootstrap samples

- Boosting
  - Wir haben wieder weak learners, dieses mal sind sie aber nicht unabhängig voneinander



↳ $\hat{c}_i(x) = \text{sgn}\left(\sum \alpha_b \cdot c_b(x)\right)$

AdaBoost.M1

Input: data $(x_1, y_1), \ldots, (x_n, y_n)$
Output: classifier $\hat{c}_B(x)$

1: **initialize** observation weights $w_i = 1/n$
2: **for** $b = 1$ to B **do**
3: Fit a classifier $c_b(x)$ to the training data using weights w_i;
4: Compute $\epsilon_b \leftarrow \sum_{i=1}^{n} w_i^{(b)} \mathbb{I}_{\{c_b(x_i) \neq y_i\}} / \sum_{i=1}^{n} w_i^{(b)}$;
5: Compute $\alpha_b \leftarrow \log \frac{1-\epsilon_b}{\epsilon_b}$;
6: Set $\forall i: \; w_i \leftarrow w_i \exp\left(\alpha_b \mathbb{I}_{\{y_i \neq c_b(x_i)\}}\right)$;
7: **end for**
8: **return** $\hat{c}_B(x) = \text{sgn}\left(\sum_{b=1}^{B} \alpha_b \, c_b(x)\right)$

- Give more weights to points, which are classified wrongly in the last classifier
  - $w_i = 1/n$ (each data-point has its own weight)
- α_i determines, how good the classifier is
  ↳ the less errors, the higher the weight

- Classifier selection
  - Train two different weak classifier types on the same subsamples
  - Compare which classifier-type performs best, choose that one
- Stumps: - Single axis partition of space
  - Choose one axis to divide space
  - Weak learner

Probability Approximately Correct Learning

- Generalization error: $R(\hat{c}) = P(\hat{c}(x) \neq c(x))$
- A machine learning algorithm is a PAC learner, if
  - Given training data $n \geq poly(1/\varepsilon, 1/\delta, dim(X))$
  - $P_{z \sim D^n}(R(\hat{c}) \leq \varepsilon) \geq 1-\delta$
    - $z \sim D^n$: sample size n
    - ε: Error
    - $1-\delta$: confidence
- General PAC learning
  - We receive sample size $n \geq poly(1/\varepsilon, 1/\delta, dim(X))$
  - $P_{z \sim D^n}(R(\hat{c}) - \inf_{c \in C} R(c) \leq \varepsilon) \geq 1-\delta$
- Finite hypothesis PAC learning
  - If the hypothesis class is finite $|H| < \infty$, then H is PAC learnable, we need
    - $n \geq 1/\varepsilon (\log|H| + \log 1/\delta)$ training samples
    - $P(R(\hat{c}) \leq \varepsilon) \geq 1-\delta$
- Efficient PAC learning algorithm
  - If it runs in polynomial time in $1/\varepsilon$ and $1/\delta$
- Examples of PAC learning
  - Axis-aligned rectangles are PAC learnable, this means, it fulfill the above requirements
  - The set of finite binary strings are not PAC learnable
- Finite classifier C PAC learning
  - If C is finite
  - $P(R(\hat{c}_n^*) - \inf R(c) > \varepsilon) \leq 2|C| \cdot \exp(-2 \cdot n \cdot \varepsilon^2)$

- VC-dimension
  - How to measure the complexity of concept class C
  - Wieviele Unterschiedliche Datapoints kann ein classifier C scattern
  - If $VC_C > 2$: $P(R(\hat{c}_n^*) - \inf R(c) > \varepsilon) \leq g_n^{VC_C} \cdot \exp(-\frac{n \cdot \varepsilon^2}{32})$
  - If C has finite VC dimension, then $P(R(\hat{c}_n^*) - \inf R(c) > \varepsilon) \leq g_n^{VC_C} \cdot \exp(-\frac{n \cdot \varepsilon^2}{32}) \xrightarrow{n \to \infty} 0$
  - Examples
    - Two points by axis-aligned rectangle
    - No set of three numbers by interval
- Strong Learners
  - Demand arbitrarily small error ε with high probability $1-\sigma$
- Weak Learners
  - PAC holds for large errors ε
  - e.g. Binary classification $\varepsilon \leq 1/2 - \gamma$
  - Used for Ensemble Methods

Useful Inequalities

- Vapnik Chervonenkis:

$$\begin{aligned}
\mathcal{R}(\hat{c}_n^\star) - \inf_{c\in\mathcal{C}} \mathcal{R}(c) &= \mathcal{R}(\hat{c}_n^\star) - \hat{\mathcal{R}}_n(\hat{c}_n^\star) + \hat{\mathcal{R}}_n(\hat{c}_n^\star) - \inf_{c\in\mathcal{C}} \mathcal{R}(c) \\
&\leq \underbrace{\mathcal{R}(\hat{c}_n^\star) - \hat{\mathcal{R}}_n(\hat{c}_n^\star)} + \underbrace{\hat{\mathcal{R}}_n(c^\star) - \mathcal{R}(c^\star)} \\
&\leq \sup_{c\in\mathcal{C}} |\hat{\mathcal{R}}_n(c) - \mathcal{R}(c)| + \sup_{c\in\mathcal{C}} |\hat{\mathcal{R}}_n(c) - \mathcal{R}(c)| \\
&\leq 2 \sup_{c\in\mathcal{C}} |\hat{\mathcal{R}}_n(c) - \mathcal{R}(c)|
\end{aligned}$$

Bound on suboptimality of $\hat{c}_n^\star$: require uniform convergence!

$$\mathbf{P}\{\mathcal{R}(\hat{c}_n^\star) - \inf_{c\in\mathcal{C}} \mathcal{R}(c) > \epsilon\} \leq \mathbf{P}\{\sup_{c\in\mathcal{C}} |\hat{\mathcal{R}}_n(c) - \mathcal{R}(c)| > \epsilon/2\}$$

- Markov: $P(X \geq \epsilon) \leq \frac{E[X]}{\epsilon}$
- Hoeffding's Inequality: $E[\exp(s \cdot X)] \leq \exp(s^2(b-a)^2/8)$
- Hoeffding's Theorem: *Let $X_1, \ldots, X_n$ be independent bounded random variables such that X_i falls in the interval $[a_i, b_i]$ with probability one and let $S_n = \sum_{i=1}^n X_i$. Then for any $t > 0$ we have*

$$\begin{aligned}
\mathbf{P}\{S_n - \mathbb{E}S_n \geq t\} &\leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right) \\
\mathbf{P}\{S_n - \mathbb{E}S_n \leq -t\} &\leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right) .
\end{aligned}$$

- Finite classifier sets: $P(\sup |\hat{R}_n(c) - R(c)| > \varepsilon) \leq 2 \cdot N \cdot \exp(-2 \cdot n \cdot \varepsilon^2)$
  - Cardinality of C bounded by N
- Risk Minimisation for Hyperplanes

Theorem 6 *Assume that X has a density. If $\hat{c}$ is found by empirical error minimization, then, for all possible distributions of (X, Y), if $n \geq d$ and $2\frac{d}{n} \leq \epsilon \leq 1$, we have*

$$\mathbf{P}\left\{\mathcal{R}(\hat{c}) > \inf_{c\in\mathcal{C}} \mathcal{R}(c) + \epsilon\right\} \leq e^{2d\epsilon} \left(2\binom{n}{d} + 1\right) \exp\left(-\frac{n\epsilon^2}{2}\right)$$

Moreover, if $n \geq d$, then

$$\mathbb{E}[\mathcal{R}(\hat{c}) - \mathcal{R}] \leq \sqrt{\frac{2}{n}(d+1)(\log n + 2)}$$

- Classifiers with vanishing errors

Theorem 7 *Assume that X has a density, and that the best linear classifier has zero error probability ($\mathcal{R}(c^\star) = 0$). Then for the ERM algorithm and for $n > d$, $\epsilon \leq 1$, it holds*

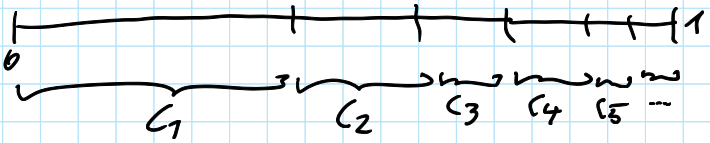
$$\mathbf{P}\{\mathcal{R}(\hat{c}_n) > \epsilon\} \leq 2\binom{n}{d} \exp(-\epsilon(n-d))$$

Nonparametric Bayesian Methods

- Number of clusters not fixed
- Dirichlet Process
  - Distribution over distribution
  - Parametrized by: α: concentration parameter
    G: Base distribution
  - Input: Distribution G } α controls the similarity
  - Output: Distribution G' }
  - Then you can draw observations from $x \sim DP(\alpha, G)$
    $\Updownarrow$
    $x \sim G'$

1. Break off sticks

- $v_1, v_2, \ldots \sim Beta(1, \alpha)$ spits out $v_i \in [0,1]$
- $c_k = v_k \cdot \prod_{j=1}^{k-1} (1 - v_k)$ (= weights)
- For each v_i we break up the stick



2. Draw atoms

- $\phi_1, \phi_2, \ldots \sim G$

3. Merge into complete distribution

- $\theta = \sum_{k \in \mathbb{N}} c_k \cdot \sigma_{\phi_k}$

  σ_{ϕ_k} is the input equivalent to $\phi_k \Rightarrow 1$ (= indicator)

- As $\alpha \to \infty$, $DP(\alpha, G) = G$
- If α is small, very few points with high probability
- If α is high, more points with high probability

- If α is ∞, $G = G'$

- The chinese Restaurant Process
  - Bunch of tables with people around it
  - New person entering the restaurant sits down at a table proportional to the number already sitting there
  - You can sit at a new table with probability proportional to α
  - Table $\hat{=}$ Cluster
  1. Take a guess initially
  2. This provides a mean for each cluster
  3. Let the number of clusters grow/shrink
  4. We want to know the assignment to a cluster for each datapoint
  5. Compute $p(z_i = k \mid z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_m, x, \alpha, G)$

$$\Leftrightarrow p(z_i = k \mid z_{-i}, x, \{\theta_k\}, \alpha)$$

(α: concentration parameter; x: Data; $\{\theta_k\}$: Means; z_{-i}: cluster assignment of other points)

- Gibbs Sampling
  - Like chinese Restaurant Process ↗ 1 cluster for every datapoint
  1. Random initialisation of assignments to clusters
  2. For iteration $i = 1, \dots$ until convergence
     - 2.1. "unassign" observation i $p(z_i \mid z_{-i})$
     - 2.2. choose new cluster for that observation
       - ↳ Either to existing one or new one

Notes

- Ridge Regression has close-term solution, Lasso doesn't
- If training error < test error => overfitting
- Feature transform of Kernel can be in ∞ dimension
- If original classifier is not consistent, Jacknive is also not consistent
- We need to add constraint $\xi \geq 0$ to soft margin
- We add the regularization term to linear regression to reduce the variance
  ↳ variance ≙ Überanpassung
- L_1 is for feature selection and better interpretability
- L_2 is for making it more robust
- The less variance we have on the prior, the more confidence we have on the prior value
- A random forest with only one tree isn't a random tree, because we sample in a random forest with bootstrapping
- Wenn separator classifier $y = 1 - x$ ist, man ihn aber als axis-oriented will, kann man ihn mit $\phi(x, y) = x + y$ transformieren
- $N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2) = N(\mu_1, \sigma_1) + N(\mu_2, \sigma_2)$

- Lasso is not strictly convex
- Ridge Regression is strictly convex
- Eine Matrix ist dann invertierbar, wenn sie vollen Rank hat $\Leftrightarrow \det(A) \neq 0$
- Closed term solution means à la linear regression inverse
- LOOCV: · Biased
  · Low variance
- MLE and MAP the same: 1. Uniform prior
  2. $n \to \infty$
- Data is linear seperable why soft margin SVM
  · Outliers give bad margin
  · Avoid overfitting
- Length l of RBNF Kernel controls, how much influence the points have further away, the larger l gets, the more information we have (= smooth lines)