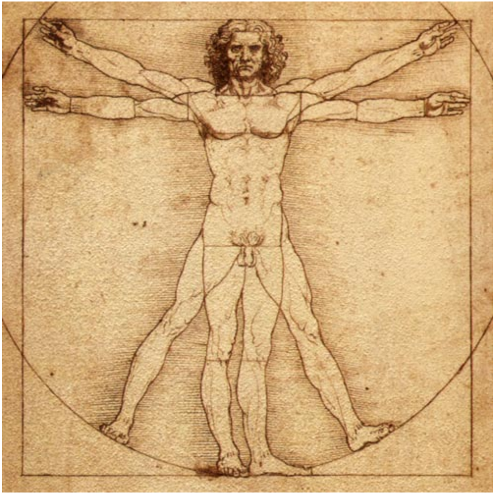
UNIVERSITY OF CAMBRIDGE

# 4F10: Ensemble Methods

Mark Gales

Michaelmas 2020

# Renaissance Man

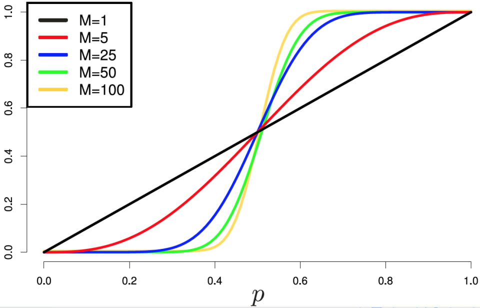


# Binary Ensemble Classifier Majority Voting

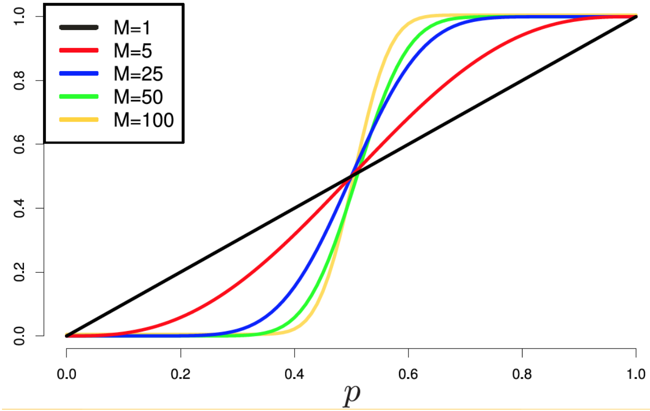
- Consider an ensemble of binary classifiers
  - each classifier's prediction is independent of other classifiers
  - select class with the most "votes" (majority voting)

$$P(\mathtt{error}) = \sum_{i=\frac{M+1}{2}}^{M} \binom{M}{i} p^i (1-p)^{M-i}$$

  - $p$ is the probability of error from a single classifier



# Ideal Binary Ensemble Error




- In practice not possible to ensure predictions independent
  - often related training data/criterion/topology

- Now consider an ensemble of discriminative classifiers
  - vary over topolgies/parameters/training data etc.

$$\hat{\boldsymbol{\omega}} = \arg\max_{\boldsymbol{\omega}} \left\{ \sum_{\{\mathcal{M}\}} \int P(\boldsymbol{\omega}|\boldsymbol{x}^\star, \boldsymbol{\theta}, \mathcal{M}) p(\boldsymbol{\theta}|\mathcal{M}, \mathcal{D}) P(\mathcal{M}|\mathcal{D}) d\boldsymbol{\theta} \right\}$$

  - $\mathcal{M}$ specifies an instance of a “model” topology
  - $\boldsymbol{\theta}$ are parameters of a particular model topology
  - highly complicated (non parametric) distribution

# Bayesian Approaches

- Now consider an ensemble of discriminative classifiers
  - vary over topolgies/parameters/training data etc.

$$\hat{\boldsymbol{\omega}} = \arg\max_{\boldsymbol{\omega}} \left\{ \sum_{\{\mathcal{M}\}} \int P(\boldsymbol{\omega}|\boldsymbol{x}^{\star}, \boldsymbol{\theta}, \mathcal{M}) p(\boldsymbol{\theta}|\mathcal{M}, \mathcal{D}) P(\mathcal{M}|\mathcal{D}) d\boldsymbol{\theta} \right\}$$

  - $\mathcal{M}$ specifies an instance of a "model" topology
  - $\boldsymbol{\theta}$ are parameters of a particular model topology
  - highly complicated (non parametric) distribution
- Use a Monte-Carlo approximation - an ensemble, $\mathcal{E}$,

$$\begin{aligned} \hat{\boldsymbol{\omega}} &= \arg\max_{\boldsymbol{\omega}} \left\{ \frac{1}{M} \sum_{j=1}^{M} P(\boldsymbol{\omega}|\boldsymbol{x}^{\star}; \boldsymbol{\theta}^{(j)}) \right\} \\ \mathcal{E} &= \left\{ \boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)} \right\}; \quad \boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta}, \mathcal{M}|\mathcal{D}); \end{aligned}$$

# Ensemble Generation

- Posterior distribution, $p(\boldsymbol{\theta}, \mathcal{M}|\mathcal{D})$, can be very complicated
  - distribution over model topologies
  - distribution over model parameters
  - distribution over model features/targets

# Ensemble Generation

- Posterior distribution, $p(\boldsymbol{\theta}, \mathcal{M}|\mathcal{D})$, can be very complicated
  - distribution over model topologies
  - distribution over model parameters
  - distribution over model features/targets
- How to sample over posterior? Two distinct elements
  - topology: network/layers/activation functions
  - parameters for a topology distribution over the parameters
- Challenging to sample (efficiently) over topologies
- Range of practical approaches to generating ensembles
  - often limited links to underlying theoretical sampling

# Bagging [1]

- Simple process based on partitioning the data:

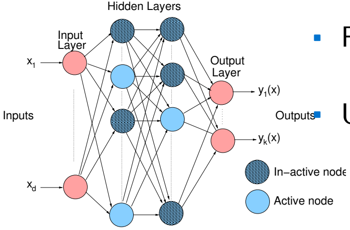
  1. from training data $\mathcal{D}$ select subset $\tilde{\mathcal{D}}$ of size $\tilde{n}$
     - selection can be with replacement
  2. train model on $\tilde{\mathcal{D}}$
     - for CML training with subset $\tilde{\mathcal{D}} = \{\{y_j, \boldsymbol{x}_j\}\}_{j=1}^{\tilde{n}}$

$$\boldsymbol{\theta}^{(i)} = \arg\max_{\boldsymbol{\theta}} \left\{ \sum_{j=1}^{\tilde{n}} \log(P(y_j|\boldsymbol{x}_j;\boldsymbol{\theta})) \right\}$$

     - each "sample" at a "local optimal"
  3. repeat until ensemble size generated

- Ensemble members built on a random sub-set of the data

# Monte Carlo Dropout [4]




- Randomly de-activate nodes during training
  - improved regularisation of network
- Usually impact of dropout "undone"
  - "deweight" weights related to dropout rate

- Ensemble generated by de-activating random nodes
  - each random selection related to draw from "local posterior"
  - topology a subset of the original topology
- Allows ensemble to trained with only a single network

# Random Network Initialisation

- For a given topology network parameters initialised used

$$\tilde{\boldsymbol{\theta}}^{(i)} \sim p(\boldsymbol{\theta}|\mathcal{M})$$

  - $p(\boldsymbol{\theta}|\mathcal{M})$ is the prior over the parameters for a topology
  - $\tilde{\boldsymbol{\theta}}^{(i)}$ is used as model initialisation
- Each member of the ensemble then be trained
  - assume CML training with supervised data $\mathcal{D} = \{\{y_j, \boldsymbol{x}_j\}\}_{j=1}^{n}$

$$\boldsymbol{\theta}^{(i)} = \arg\max_{\boldsymbol{\theta}} \left\{ \sum_{j=1}^{n} \log(P(y_j|\boldsymbol{x}_j, \tilde{\boldsymbol{\theta}}^{(i)}; \boldsymbol{\theta})) \right\}$$

  - each "sample" at a "local optimal"

# “Adhoc” Models

- Simple approach: select different configurations:
  1. number/size of layers
  2. nature of activation function/training cost function
  3. nature of targets

- Manually “sampling” from the “configuration posterior”
  - standard approaches used in many systems
  - each system at a local optimum
  - allows rich diversity of models to be used
  - expert knowledge to make systems complementary

# Posterior Combination

- Simplest approach: average class posteriors from the ensemble

$$P(\omega|\boldsymbol{x}^\star;\mathcal{E}) = \frac{1}{M}\sum_{i=1}^{M} P(\omega|\boldsymbol{x}^\star;\boldsymbol{\theta}^{(i)})$$

  - requires all models in the ensemble to be evaluated
  - requires all model in the ensemble to be stored
  - no explicit measure of structure in the selection

# Posterior Combination

- Simplest approach: average class posteriors from the ensemble

$$P(\omega|\boldsymbol{x}^\star;\mathcal{E}) = \frac{1}{M}\sum_{i=1}^{M} P(\omega|\boldsymbol{x}^\star;\boldsymbol{\theta}^{(i)})$$

  - requires all models in the ensemble to be evaluated
  - requires all model in the ensemble to be stored
  - no explicit measure of structure in the selection

- Interesting to consider:
  - can the ensemble be compressed to a single model?

# Model Compression: Teacher-Student Training [2]

- Consider cross entropy training criterion, supervised training
  - training data $\mathcal{D} = \{\{\boldsymbol{x}_1, y_1\}, \ldots, \{\boldsymbol{x}_n, y_n\}\}$, $y_i \in \{\omega_1, \ldots, \omega_K\}$

$$\mathcal{F}_{\text{ce}} = -\sum_{i=1}^{n} \log\left(P(y_i|\boldsymbol{x}_i; \boldsymbol{\theta}_{\text{S}})\right) = -\sum_{i=1}^{n} \sum_{\omega} \delta(y_i, \omega) \log\left(P(\omega|\boldsymbol{x}_i; \boldsymbol{\theta}_{\text{S}})\right)$$

  - $\delta(y_i, \omega)$ a Kronecker delta function, sum over all classes

# Model Compression: Teacher-Student Training [2]

- Consider cross entropy training criterion, supervised training
  - training data $\mathcal{D} = \{\{\boldsymbol{x}_1, y_1\}, \ldots, \{\boldsymbol{x}_n, y_n\}\}$, $y_i \in \{\omega_1, \ldots, \omega_K\}$

$$\mathcal{F}_{\texttt{ce}} = -\sum_{i=1}^{n} \log\left(P(y_i|\boldsymbol{x}_i; \boldsymbol{\theta}_{\mathrm{S}})\right) = -\sum_{i=1}^{n}\sum_{\omega} \delta(y_i, \omega) \log\left(P(\omega|\boldsymbol{x}_i; \boldsymbol{\theta}_{\mathrm{S}})\right)$$

  - $\delta(y_i, \omega)$ a Kronecker delta function, sum over all classes
- Modify targets based on a teacher

$$\mathcal{F}_{\texttt{ts}} = -\sum_{i=1}^{n}\sum_{\omega} P(\omega|\boldsymbol{x}_i; \boldsymbol{\theta}_{\mathrm{T}}) \log\left(P(\omega|\boldsymbol{x}_i; \boldsymbol{\theta}_{\mathrm{S}})\right)$$

  - "soft" targets rather than "hard" targets to train $\boldsymbol{\theta}_{\mathrm{S}}$
  - targets obtained from teacher network, $\boldsymbol{\theta}_{\mathrm{T}}$
- Also called model distillation

# Teacher-Student Ensemble Combination

- Originally used for model compression
  - student network, $\boldsymbol{\theta}_{\mathtt{S}}$, simpler than teacher network, $\boldsymbol{\theta}_{\mathtt{T}}$
  - can also be used for ensemble combination
- Replace a single teacher by an ensemble, $\mathcal{E}$, now

$$\mathcal{F}_{\mathtt{ts}} = -\sum_{i=1}^{n} \sum_{\omega} P(\omega|\boldsymbol{x}_i; \mathcal{E}) \log \left( P(\omega|\boldsymbol{x}_i; \boldsymbol{\theta}_{\mathtt{S}}) \right)$$

- Various forms possible for targets, for example

$$P(\omega|\boldsymbol{x}_i; \mathcal{E}) = \frac{1}{M} \sum_{j=1}^{M} P(\omega|\boldsymbol{x}_i; \boldsymbol{\theta}^{(j)})$$

  - where there are $M$ models in the ensemble $\mathcal{E}$

# Bias Variance Trade-Off (Reference Only)

# Bias Variance Trade-Off - Regression

- Consider the variance of a trained regression model $\hat{f}(\boldsymbol{x})$
  - the ideal regression model is $f(\boldsymbol{x})$: $y = f(\boldsymbol{x}) + \epsilon$
  - supervised training examples: $\mathcal{D} = \{\{\boldsymbol{x}_1, y_1\}, \dots, \{\boldsymbol{x}_n, y_n\}\}$
  - use data to train regression model $\hat{f}(\boldsymbol{x})$

$$
\begin{aligned}
\mathbb{E}\left\{(\hat{f}(\boldsymbol{x}) - y)^2\right\} &= \mathbb{E}\left\{\epsilon^2\right\} - (\mathbb{E}\left\{\epsilon\right\})^2 + \mathbb{E}\left\{\hat{f}(\boldsymbol{x})^2\right\} - \left(\mathbb{E}\left\{\hat{f}(\boldsymbol{x})\right\}\right)^2 \\
&\quad + \left(\mathbb{E}\left\{f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\right\}\right)^2 \\
&= \mathrm{Var}(\epsilon) + \mathrm{Var}(\hat{f}(\boldsymbol{x})) + (\mathrm{Bias}(\hat{f}(\boldsymbol{x})))^2
\end{aligned}
$$

  - expectation is over all sets of training data drawn from $p(\boldsymbol{x}, y)$

# Bias Variance Trade-Off - Regression

- Consider the variance of a trained regression model $\hat{f}(\boldsymbol{x})$
  - the ideal regression model is $f(\boldsymbol{x})$: $y = f(\boldsymbol{x}) + \epsilon$
  - supervised training examples: $\mathcal{D} = \{\{\boldsymbol{x}_1, y_1\}, \dots, \{\boldsymbol{x}_n, y_n\}\}$
  - use data to train regression model $\hat{f}(\boldsymbol{x})$

$$\begin{aligned}
\mathbb{E}\left\{(\hat{f}(\boldsymbol{x}) - y)^2\right\} &= \mathbb{E}\left\{\epsilon^2\right\} - (\mathbb{E}\{\epsilon\})^2 + \mathbb{E}\left\{\hat{f}(\boldsymbol{x})^2\right\} - \left(\mathbb{E}\left\{\hat{f}(\boldsymbol{x})\right\}\right)^2 \\
&\quad + \left(\mathbb{E}\left\{f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\right\}\right)^2 \\
&= \mathrm{Var}(\epsilon) + \mathrm{Var}(\hat{f}(\boldsymbol{x})) + (\mathrm{Bias}(\hat{f}(\boldsymbol{x})))^2
\end{aligned}$$

  - expectation is over all sets of training data drawn from $p(\boldsymbol{x}, y)$
- Another view of generalisation:
  - note: can't do anything about noise in the data

| | | | | |
|---|---|---|---|---|
| improved generalisation | $\rightarrow$ | simpler model | $\rightarrow$ | higher bias |
| improved data modelling | $\rightarrow$ | complex model | $\rightarrow$ | higher variance |

# Ensemble Bias Variance Trade-Off (Regression)

- Consider a set of training data sets: $\mathcal{D}_1, \ldots, \mathcal{D}_N$
  - train regressor for each training set: $\hat{f}^{(1)}(\boldsymbol{x}), \ldots, \hat{f}^{(N)}(\boldsymbol{x})$
  - ensemble prediction

$$\hat{f}(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \hat{f}^{(i)}(\boldsymbol{x})$$

- The expected prediction error is given by (true target $y$)

$$\mathbb{E}\left\{(\hat{f}(\boldsymbol{x}) - y)^2\right\} = \mathrm{Var}(\epsilon) + \frac{1}{N}\mathrm{Var}(\hat{f}^{(i)}(\boldsymbol{x})) + \left(1 - \frac{1}{N}\right)\mathrm{Cov}(\hat{f}^{(i)}(\boldsymbol{x})) + (\mathrm{Bias}(\hat{f}^{(i)}(\boldsymbol{x})))^2$$

  - $\mathrm{Var}(\hat{f}^{(i)}(\boldsymbol{x}))$: ind. class. var. $\mathbb{E}\left\{(\hat{f}^{(i)}(\boldsymbol{x}))^2\right\} - \mathbb{E}\left\{(\hat{f}^{(i)}(\boldsymbol{x}))\right\}^2$
  - $\mathrm{Bias}(\hat{f}^{(i)}(\boldsymbol{x}))$: bias of individual classifier $\mathbb{E}\left\{f(\boldsymbol{x}) - \hat{f}^{(i)}(\boldsymbol{x})\right\}$
  - Cov: covariance between classifiers $\mathbb{E}\{(f^{(i)}(\boldsymbol{x}) - f^{(j)}(\boldsymbol{x}))^2\}$

# Adaboost (Reference)

# Boosting (AdaBoost) [3]

- General approach for generating/combining an ensemble
  - converts multiple weak learners to strong learners
  - form described here is AdaBoost

- Consider a 2-class classification problem - training data

$$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}, \quad y_i \in \{-1, 1\}$$

- Combine classifiers as a recursive linear combination

$$\overline{\mathcal{F}}_m(\boldsymbol{x}_i) = \overline{\mathcal{F}}_{m-1}(\boldsymbol{x}_i) + \alpha_m \mathcal{F}(\boldsymbol{x}_i; \boldsymbol{\theta}^{(m)})$$

- How to train classifier $\mathcal{F}(\boldsymbol{x}_i; \boldsymbol{\theta}^{(m)})$ how to get weight $\alpha_m$?

# AdaBoost (1)

- Define total cost function to minimise $E(\boldsymbol{\theta}^{(m)})$
  - for correct classification $y_i = \mathcal{F}(\boldsymbol{x}_i; \boldsymbol{\theta}^{(m)})$, $y_i\mathcal{F}(\boldsymbol{x}_i; \boldsymbol{\theta}^{(m)}) = 1$
  - cost function to find $\boldsymbol{\theta}^{(m)}$ - minimise

$$
\begin{aligned}
E(\boldsymbol{\theta}) &= \sum_{i=1}^{n} \exp\left(-y_i \overline{\mathcal{F}}_m(\boldsymbol{x}_i)\right) \\
&= \sum_{i=1}^{n} \mathcal{G}_{m-1}(\boldsymbol{x}_i) \exp\left(-y_i \alpha_m \mathcal{F}(\boldsymbol{x}_i; \boldsymbol{\theta})\right)
\end{aligned}
$$

- Re-express criterion based on correct/incorrect using $\mathcal{F}(\boldsymbol{x}_i; \boldsymbol{\theta})$
  - simple to show that

$$
\begin{aligned}
E(\boldsymbol{\theta}) = & \sum_{i=1}^{n} \mathcal{G}_{m-1}(\boldsymbol{x}_i) \exp(-\alpha_m) \\
& + \sum_{y_i \neq \mathcal{F}(\boldsymbol{x}_i; \boldsymbol{\theta})} \mathcal{G}_{m-1}(\boldsymbol{x}_i) \left(\exp(\alpha_m) - \exp(-\alpha_m)\right)
\end{aligned}
$$

# AdaBoost (2)

- Following on from previous cost function expression
  - optimal additional binary classifier, $\boldsymbol{\theta}^{(m)}$, minimises

$$
\begin{aligned}
\boldsymbol{\theta}^{(m)} &= \arg\min_{\boldsymbol{\theta}} \left\{ \sum_{y_i \neq \mathcal{F}(\boldsymbol{x}_i;\boldsymbol{\theta})} \mathcal{G}_{m-1}(\boldsymbol{x}_i) \right\} \\
&= \arg\min_{\boldsymbol{\theta}} \left\{ \sum_{y_i \neq \mathcal{F}(\boldsymbol{x}_i;\boldsymbol{\theta})} \exp\left(-y_i \overline{\mathcal{F}}_{m-1}(\boldsymbol{x}_i)\right) \right\}
\end{aligned}
$$

  - can show that optimal weights are

$$
\alpha_m = \frac{1}{2} \log \left( \frac{\sum_{y_i = \mathcal{F}(\boldsymbol{x}_i;\boldsymbol{\theta}^{(m)})} \mathcal{G}_{m-1}(\boldsymbol{x}_i)}{\sum_{y_i \neq \mathcal{F}(\boldsymbol{x}_i;\boldsymbol{\theta}^{(m)})} \mathcal{G}_{m-1}(\boldsymbol{x}_i)} \right)
$$

- Has been extended to multiple classes

[1] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[2] C. Bucila, R. Caruana, and A. Niculescu-Mizil, “Model compression,” in *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, 2006, pp. 535–541.

[3] R. E. Schapire, “A brief introduction to boosting,” in *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 1401–1406. [Online]. Available: http://dl.acm.org/citation.cfm?id=1624312.1624417

[4] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.