MA4270 Data Modelling and Computation

Preliminaries

- **Variance** $= \mathbb{E}[(X-\mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$
- **Markov ineq.**: Given $a > 0$ and X nonnegative, $\Pr(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$
- **Chebyshev ineq.**: $\Pr(|Y - \mathbb{E}[Y]| \geq b) \leq \frac{\sigma_Y^2}{b^2}$

Binary Classification

- **Training data**: $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$
  - where $y_t \in \{-1, 1\}$ and $\mathbf{x}_t \in \mathbb{R}^d$
  - where we have n training data
- **Classifier**: $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \{-1, 1\}$
- **Training error**: $\hat{E}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{t=1}^n \mathrm{Loss}(y_t, f_{\boldsymbol{\theta}}(\mathbf{x}_t))$
  - i.e. the proportion of failed training data
- $\mathrm{Loss}(y, \hat{y}) = \mathbf{1}\{\hat{y} \neq y\}$

Linear classifier

- **Formula**: $\hat{y} := \mathrm{sgn}\langle \boldsymbol{\theta}, \mathbf{x}\rangle := \mathrm{sgn}\sum_{i=1}^d \theta_i x_i$ (for some fixed $\boldsymbol{\theta}$
  - i.e. partitioning space with plane passing through origin
- $\mathcal{D}$ is linearly separable: $\exists \boldsymbol{\theta}$ such that $\hat{E}(\boldsymbol{\theta}) = 0$
- **Perceptron update algorithm**:
  - current estimate $\boldsymbol{\theta}_{curr}$
  - look at one $(\mathbf{x}_t, y_t)$
  - if $\mathrm{sgn}\langle \boldsymbol{\theta}_{curr}, \mathbf{x}_t\rangle = y_t$ then $\boldsymbol{\theta}_{next} := \boldsymbol{\theta}_{curr}$
  - otherwise set $\boldsymbol{\theta}_{next} := \boldsymbol{\theta}_{curr} + y_t\mathbf{x}_t$

  (intuitively, we are increasing the value of $y_t\boldsymbol{\theta}_{next}^T\mathbf{x}_t$)
- **Perceptron algorithm**:
  - init $\boldsymbol{\theta}^{(0)} := \mathbf{0}, k := 0$
  - For each data point, run the update, and increment k only when we made a mistake (repeat after reaching n)
  - stop when we do full pass of all data without mistakes
- **Perceptron algorithm assumptions**:
  - $\exists R > 0$ such that $\forall t, \|\mathbf{x}_t\| \leq R$ (i.e. $\mathbf{x}_t$ is uniformly bdd)
  - $\exists \boldsymbol{\theta}^*$ and $\exists \gamma > 0$ such that $\min_t y_t(\boldsymbol{\theta}^*)^T\mathbf{x}_t \geq \gamma$ (i.e. stronger lin. separable cond. – bounded away from zero)

  Thm: under above assumptions, the algorithm finds $\boldsymbol{\theta}^{(k)}$ such that $\hat{E}(\boldsymbol{\theta}^{(k)}) = 0$ after at most $k_{max} = \frac{R^2\|\boldsymbol{\theta}^*\|^2}{\gamma^2}$ steps (we make at most k_{max} mistakes)
- **Cauchy-Schwarz inequality**: $|\langle \mathbf{u}, \mathbf{v}\rangle| \leq \|\mathbf{u}\| \cdot \|\mathbf{v}\|$ (with equality when $\mathbf{u}$ and $\mathbf{v}$ are in the same direction)
- **Perceptron proof**:
  1. show that $\langle \boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k+1)}\rangle \geq \langle \boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k)}\rangle + \gamma$, so $\langle \boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k)}\rangle \geq k\gamma$
  2. show that $\left\|\boldsymbol{\theta}^{(k+1)}\right\|^2 \leq \left\|\boldsymbol{\theta}^{(k)}\right\|^2 + R^2$, so $\left\|\boldsymbol{\theta}^{(k)}\right\|^2 \leq kR^2$
  3. use Cauchy-Schwartz ineq. to conclude that $k \leq \frac{R^2\|\boldsymbol{\theta}^*\|^2}{\gamma^2}$

  Details:
  1. $(\boldsymbol{\theta}^*)^T\boldsymbol{\theta}^{(k+1)} = (\boldsymbol{\theta}^*)^T(\boldsymbol{\theta}^{(k)} + y_t\mathbf{x}_t) = (\boldsymbol{\theta}^*)^T\boldsymbol{\theta}^{(k)} + y_t(\boldsymbol{\theta}^*)^T\mathbf{x}_t \geq (\boldsymbol{\theta}^*)^T\boldsymbol{\theta}^{(k)} + \gamma$
  2. $\|\boldsymbol{\theta}^{(k+1)}\|^2 = \|\boldsymbol{\theta}^{(k)} + y_t\mathbf{x}_t\|^2 = \|\boldsymbol{\theta}^{(k)}\|^2 + 2\langle\boldsymbol{\theta}^{(k)}, y_t\mathbf{x}_t\rangle + \|\mathbf{x}_t\|^2 \leq \|\boldsymbol{\theta}^{(k)}\|^2 + \|\mathbf{x}_t\|^2$ (since when a mistake occurs we must have $\langle\boldsymbol{\theta}^{(k)}, y_t\mathbf{x}_t\rangle \leq 0$)
  3. $1 \geq \frac{\langle\boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k)}\rangle}{\|\boldsymbol{\theta}^*\|\cdot\|\boldsymbol{\theta}^{(k)}\|} \geq \frac{k\gamma}{\|\boldsymbol{\theta}^*\|\cdot\sqrt{kR^2}} = \frac{\sqrt{k}\gamma}{\|\boldsymbol{\theta}^*\|\cdot R}$, so $k \leq \frac{R^2\|\boldsymbol{\theta}^*\|^2}{\gamma^2}$
- **Perceptron with offset**:

  Set $\hat{\boldsymbol{\theta}} := \begin{bmatrix}\boldsymbol{\theta} \\ \theta_0\end{bmatrix}$ and $\hat{\mathbf{x}} := \begin{bmatrix}\mathbf{x} \\ 1\end{bmatrix}$,

  so that $\hat{\boldsymbol{\theta}}^T\hat{\mathbf{x}} = \boldsymbol{\theta}^T\mathbf{x} + \theta_0$ and $\hat{R}^2 = R^2 + c^2$
- **(Non-geometric) margin** (w.r.t. $\boldsymbol{\theta}^*$): $\gamma := \min_t y_t(\boldsymbol{\theta}^*)^T\mathbf{x}_t$ (note: usually "margin" means the geometric margin)
- **Geometric margin** (normalised by $\|\boldsymbol{\theta}^*\|$): $\gamma_{geom} := \frac{\gamma}{\|\boldsymbol{\theta}^*\|}$ (= the shortest distance from a point to the hyperplane)
- **Maximum margin classifier** (a kind of support vector machine): find the classifier that makes the margin largest
  - maximise (over $\boldsymbol{\theta}$, γ) $\frac{\gamma}{\|\boldsymbol{\theta}\|}$ subject to $\forall t$, $y_t\boldsymbol{\theta}^T\mathbf{x}_t \geq \gamma$
  - equiv. minimise (over new $\boldsymbol{\theta}$) $\frac{1}{2}\|\boldsymbol{\theta}\|^2$ subject to $\forall t$, $y_t\boldsymbol{\theta}^T\mathbf{x}_t \geq 1$
  - which is a convex optimisation problem solvable efficiently
  - then the classifier $f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathrm{sgn}\left(\boldsymbol{\theta}^T\mathbf{x}\right)$ is max margin
  - and $\gamma_{geom} = \frac{1}{\|\boldsymbol{\theta}^*\|}$ (new $\boldsymbol{\theta}$)

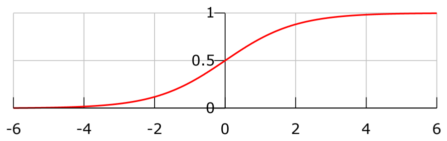
  Proof of uniqueness: can show that if we have distinct $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$ both optimal, then their average most also be optimal and hence $\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|^2 = 0$
- **Support vector**: the $\mathbf{x}$s that touch the margin

Linear classifier with offset

- **Formula**: $\hat{y} := \mathrm{sgn}(\langle\boldsymbol{\theta}, \mathbf{x}\rangle + \theta_0)$ where $\boldsymbol{\theta} \in \mathbb{R}^d$ and $\theta_0 \in \mathbb{R}$
- **SVM with offset**: minimise (over $\boldsymbol{\theta}$, θ_0) $\frac{1}{2}\|\boldsymbol{\theta}\|^2$ subject to $\forall t$, $y_t\left(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0\right) \geq 1$ (note: we don't penalise for θ_0)
- **Soft-margin SVM**: allows misclassified points by introducing slack variables ($\boldsymbol{\zeta}$):
  - minimise (over $\boldsymbol{\theta}$, θ_0, $\boldsymbol{\zeta}$) $\frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{t=1}^n \zeta_t$ subject to $\forall t$, $y_t\left(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0\right) \geq 1 - \zeta_t$ and $\forall t$, $\zeta_t \geq 0$ (allows us to pay a measured penalty for points too close to the margin or misclassified)
  - equiv. minimise (over $\boldsymbol{\theta}$, θ_0) $\frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{t=1}^n \left[1 - y_t\left(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0\right)\right]_+$ where $[z]_+ := \max\{0, z\}$
  - higher $C \implies$ favour fewer violations
  - $C \to \infty$ and $\mathcal{D}$ is lin. separable $\implies$ hard-margin SVM

  Support vector: the $\mathbf{x}$s that touch the margin, are inside the margin, or misclassified
  - moving these points will change the line
  - if we remove all points except those from support vector, the classifier will remain the same
- Loss: (where $z = y_t\left(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0\right)$)
  - 0-1 loss: $Loss(z) := \mathbf{1}\{z \leq 0\}$
  - Hinge loss: $Loss_h(z) := [1 - z]_+$ (it is convex)

Logistic regression



- **Logistic function**: $g(z) := \frac{1}{1+e^{-z}}$
- **Soft decision**: We answer using a probability space in $[0, 1]$ (logistic regression is a soft-decision algorithm)
- **Probability** (given $\boldsymbol{\theta}$, θ_0):

  $P(y = 1 \mid \mathbf{x}) = g\left(\boldsymbol{\theta}^T\mathbf{x} + \theta_0\right) = \frac{1}{1+\exp\left(-\left(\boldsymbol{\theta}^T\mathbf{x}+\theta_0\right)\right)}$

  $P(y = -1 \mid \mathbf{x}) = 1 - P(y = 1 \mid \mathbf{x}) = \frac{1}{1+\exp\left(\boldsymbol{\theta}^T\mathbf{x}+\theta_0\right)}$

  Hence $P(y \mid \mathbf{x}) = g\left(y \cdot \left(\boldsymbol{\theta}^T\mathbf{x} + \theta_0\right)\right)$

  Note: $\mathrm{Log}\left(\frac{P(y=1|\mathbf{x})}{P(y=-1|\mathbf{x})}\right) = \boldsymbol{\theta}^T\mathbf{x} + \theta_0$
- Multiplying $\boldsymbol{\theta}$ and θ_0 by a large constant makes the probability further away from 0.5 (i.e. the graph gets steeper, but the decision boundary remains the same)
- **To determine $\boldsymbol{\theta}$ and θ_0**, we generally use "maximum likelihood":

  $L(\boldsymbol{\theta}, \theta_0 \mid \mathcal{D}) = \prod_{t=1}^n P(y_t \mid \mathbf{x}_t; \boldsymbol{\theta}, \theta_0)$

  (generally good enough if n is large)

  Equiv. $L(\boldsymbol{\theta}, \theta_0 \mid \mathcal{D}) = P(y_1, \ldots, y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n; \boldsymbol{\theta}, \theta_0)$

  max likelihood: $(\boldsymbol{\theta}, \theta_0) = \arg\max_{\boldsymbol{\theta}, \theta_0} L(\boldsymbol{\theta}, \theta_0 \mid \mathcal{D})$
  - To solve:

  $\arg\max_{\boldsymbol{\theta}, \theta_0} L(\boldsymbol{\theta}, \theta_0 \mid \mathcal{D})$
  $= \arg\max_{\boldsymbol{\theta}, \theta_0} \prod_{t=1}^n P(y_t \mid \mathbf{x}_t; \boldsymbol{\theta}, \theta_0)$
  $= \arg\max_{\boldsymbol{\theta}, \theta_0} \sum_{t=1}^n \mathrm{Log}\, P(y_t \mid \mathbf{x}_t; \boldsymbol{\theta}, \theta_0)$
  $= \arg\min_{\boldsymbol{\theta}, \theta_0} \sum_{t=1}^n \mathrm{Log}(1 + \exp(-y_t(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0)))$

  ... which can only be solved numerically using gradient descent, because it is convex
  - We can call it a "loss":

  Logistic loss: $Loss(z) = \mathrm{Log}(1 + e^{-z})$
- **Gradient descent for max likelihood logistic regression**:

  $\frac{\partial}{\partial\theta_0}\mathrm{Log}(1 + \exp(-y_t(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0))) = -y_t\left(1 - P(y_t \mid \mathbf{x}_t; \boldsymbol{\theta}, \theta_0)\right)$

  $\frac{\partial}{\partial\boldsymbol{\theta}}\mathrm{Log}(1 + \exp(-y_t(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0))) = -y_t\mathbf{x}_t\left(1 - P(y_t \mid \mathbf{x}_t; \boldsymbol{\theta}, \theta_0)\right)$
  - Thus:

  $\theta_0^{(i+1)} \leftarrow \theta_0^{(i)} + \eta\sum_{i=1}^n y_t\left(1 - P\left(y_t \mid \mathbf{x}_t; \boldsymbol{\theta}^{(n)}, \theta_0^{(n)}\right)\right)$

  $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)} + \eta\sum_{i=1}^n y_t\mathbf{x}_t\left(1 - P\left(y_t \mid \mathbf{x}_t; \boldsymbol{\theta}^{(n)}, \theta_0^{(n)}\right)\right)$
- **Stochastic gradient descent**:
  - For large data sets, iterating all points at every step is too costly, so we instead pick one (or a few) data points only (at random, or cycle, etc):

  $\theta_0^{(i+1)} \leftarrow \theta_0^{(i)} + \eta y_t\left(1 - P\left(y_t \mid \mathbf{x}_t; \boldsymbol{\theta}^{(n)}, \theta_0^{(n)}\right)\right)$

  $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)} + \eta y_t\mathbf{x}_t\left(1 - P\left(y_t \mid \mathbf{x}_t; \boldsymbol{\theta}^{(n)}, \theta_0^{(n)}\right)\right)$
- For separable data, max likelihood will lead to an arbitrarily steep logistic regression function (i.e. $L(\boldsymbol{\theta}, \theta_0 \mid \mathcal{D}) \to 1$ as $\boldsymbol{\theta}, \theta_0 \to \infty$ is the max likelihood), so we want to prevent this using regularisation
- **Regularisation**: Change the expression to $\arg\min_{\boldsymbol{\theta}, \theta_0} \sum_{t=1}^n \mathrm{Log}(1 + \exp(-y_t(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0))) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$ (where λ is the regularisation parameter) (prevents overly large $\boldsymbol{\theta}$)
- Logistic regression can be extended to multi-class classification

Gradient descent

- **Def**: Want to find $\hat{\mathbf{z}} = \arg\min_{\mathbf{z}\in\mathbb{R}^d} f(\mathbf{z})$ for some evaluation function f
- **Algorithm**: At the current point, evaluate the current gradient, and move a bit in the steepest downhill direction
  - update operation: $\mathbf{z}^{(i+1)} \leftarrow \mathbf{z}^{(i)} - \eta\nabla f(\mathbf{z}^{(i)})$

  where $\nabla f(\mathbf{z}^{(i)}) := \begin{bmatrix}\frac{\partial f}{\partial z_1} \\ \vdots \\ \frac{\partial f}{\partial z_n}\end{bmatrix}$ and η is the step size
  - guaranteed to find the minimum if the function is convex

Linear regression

- The output is now over $\mathbb{R}$
- **Training data**: $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$
  - where $y_t \in \mathbb{R}$ and $\mathbf{x}_t \in \mathbb{R}^d$
- **Objective**: To learn a prediction rule (line): $\hat{y} = \boldsymbol{\theta}^T\mathbf{x} + \theta_0$
- **Gaussian model**: Add some noise, because the relationship may not be perfect
  - $y = (\boldsymbol{\theta}^*)^T\mathbf{x} + \theta_0^* + z$, where $z \sim N(0, \sigma^2)$
  - PDF of a normal distribution: $\mathcal{N}(z; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(z-\mu)^2}{2\sigma^2}\right)$
  - So $P(y \mid \mathbf{x}) = \mathcal{N}\left(y; (\boldsymbol{\theta}^*)^T\mathbf{x} + \theta_0^*, \sigma^2\right)$
- **Using maximum likelihood to derive least squares formula** (works for Gaussian noise model only):

  $L(\boldsymbol{\theta}, \theta_0, \sigma^2 \mid \mathcal{D})$
  $= P(y_1, \ldots, y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n; \boldsymbol{\theta}, \theta_0, \sigma^2)$
  $= \prod_{t=1}^n P(y_t \mid \mathbf{x}_t; \boldsymbol{\theta}, \theta_0, \sigma^2)$
  $= \prod_{t=1}^n \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\left(y_t - \boldsymbol{\theta}^T\mathbf{x}_t - \theta_0\right)^2}{2\sigma^2}\right)$

  $\mathrm{Log}(L(\boldsymbol{\theta}, \theta_0, \sigma^2 \mid \mathcal{D}))$
  $= -\frac{n}{2}\mathrm{Log}(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{t=1}^n\left(y_t - \boldsymbol{\theta}^T\mathbf{x}_t - \theta_0\right)^2$

  ... so $\boldsymbol{\theta}$ and θ_0 do not depend on σ^2

  ... so equiv. to finding

  $\left(\hat{\boldsymbol{\theta}}, \hat{\theta}_0\right) = \arg\min_{\boldsymbol{\theta}, \theta_0}\sum_{t=1}^n\left(y_t - \left(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0\right)\right)^2$

  (note that $\left(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0\right)$ is simply the predicted value of y using the line defined by $\left(\hat{\boldsymbol{\theta}}, \hat{\theta}_0\right)$, so it is the formula for least squares regression)
- **Closed form solution** (for the prediction/estimate):

  Solve $J(\boldsymbol{\Theta}) := \sum_{t=1}^n\left(y_t - \left(\boldsymbol{\theta}^T\mathbf{x}_t + \theta_0\right)\right)^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\Theta}\|^2$, where:
  - $\mathbf{y} = \begin{bmatrix}y_1 \\ \vdots \\ y_n\end{bmatrix} \in \mathbb{R}^n$; $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1^T & 1 \\ \vdots & \vdots \\ \mathbf{x}_n^T & 1\end{bmatrix} \in \mathbb{R}^{n\times(d+1)}$; $\boldsymbol{\Theta} = \begin{bmatrix}\boldsymbol{\theta} \\ \theta_0\end{bmatrix} \in \mathbb{R}^{d+1}$

  then to solve for min point, we make the gradient equal 0:

  $\nabla\left(\|\mathbf{y} - \mathbf{X}\boldsymbol{\Theta}\|^2\right) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\Theta}) = \mathbf{0}$

  $\implies \mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\boldsymbol{\Theta}$

  $\implies \boldsymbol{\Theta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$ (so $\boldsymbol{\Theta}$ is linear in $\mathbf{y}$ but not $\mathbf{X}$)

  ($\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$ is called the pseudo-inverse of $\mathbf{X}$)

  Prediction rule: $\hat{y} = \hat{\boldsymbol{\theta}}^T\mathbf{x} + \hat{\theta}_0$
- Maximum likelihood with high n: low bias and low variance
  Maximum likelihood with low n: low bias and high variance

Bias & Variance

- **Bias**: systematic error
  Variance: random error

How good is our predicted $\boldsymbol{\Theta}$?

- **Goal**: minimise $\mathbb{E}\left[\left\|\hat{\boldsymbol{\Theta}}-\boldsymbol{\Theta}^*\right\|^2\right]$ (i.e. mean squared error)
  (note: $\hat{\boldsymbol{\Theta}}$ is our estimate; $\boldsymbol{\Theta}^*$ is actual)
- **Bias-variance decomposition**:
  $$\underbrace{\mathbb{E}\left[\left\|\hat{\boldsymbol{\Theta}}-\boldsymbol{\Theta}^*\right\|^2\right]}_{\text{mean sq. error}} = \underbrace{\left\|\mathbb{E}\left[\hat{\boldsymbol{\Theta}}\right]-\boldsymbol{\Theta}^*\right\|^2}_{\text{bias squared}} + \underbrace{\mathbb{E}\left[\left\|\hat{\boldsymbol{\Theta}}-E\left[\hat{\boldsymbol{\Theta}}\right]\right\|^2\right]}_{\text{variance}}$$
  Note: $\mathbb{E}\left[\hat{\boldsymbol{\Theta}}\right]-\boldsymbol{\Theta}^* = -\lambda\left(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}\right)^{-1}\boldsymbol{\Theta}^*$
- **To show that least squares estimate is unbiased**
  - Combining all data points for a true value ($\boldsymbol{\Theta}^*$): $y=(\boldsymbol{\theta}^*)^T\mathbf{x}+\theta_0^*+z$ into a vector, we get $\mathbf{y}=\mathbf{X}\boldsymbol{\Theta}^*+\mathbf{z}$
  - substitute equation into $\hat{\boldsymbol{\Theta}}=\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$ to get $\hat{\boldsymbol{\Theta}}=\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\left(\mathbf{X}\boldsymbol{\Theta}^*+\mathbf{z}\right)=\boldsymbol{\Theta}^*+\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{z}$
  - and since $\mathbb{E}[\mathbf{z}]=0$, we have $\mathbb{E}\left[\hat{\boldsymbol{\Theta}}\right]=\boldsymbol{\Theta}^*$
  - so we conclude that $\hat{\boldsymbol{\Theta}}$ is an unbiased estimate for $\boldsymbol{\Theta}^*$
- **Variance term** may be very high if $\mathbf{X}^T\mathbf{X}$ has small eigenvalues (intuitively, $\mathbf{X}^T\mathbf{X}$ is "almost singular")
  $\mathbb{E}\left[\left\|\hat{\boldsymbol{\Theta}}-E\left[\hat{\boldsymbol{\Theta}}\right]\right\|^2\right]=\operatorname{Tr}\left[\operatorname{Cov}\left[\hat{\boldsymbol{\Theta}}\right]\right]$, and
  $\operatorname{Cov}\left[\hat{\boldsymbol{\Theta}}\right]=\sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$, so
  $\mathbb{E}\left[\left\|\hat{\boldsymbol{\Theta}}-E\left[\hat{\boldsymbol{\Theta}}\right]\right\|^2\right]=\sigma^2\operatorname{Tr}\left[\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right]$
  ... and thus variance high when $\mathbf{X}^T\mathbf{X}$ has small eigenvalues
- **Trading bias for variance – Regularisation** (with $\frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$) (aka. "ridge regression"):
  - we want to penalize large $\boldsymbol{\theta}$ (but not θ_0)
  $\left(\hat{\boldsymbol{\theta}},\hat{\theta}_0\right)$
  $=\arg\min_{\boldsymbol{\theta},\theta_0}\sum_{t=1}^n\left(y_t-\left(\boldsymbol{\theta}^T\mathbf{x}_t+\theta_0\right)\right)^2+\frac{\lambda}{2}\sum_{j=1}^d\theta_j^2$
  $=\arg\min_{\boldsymbol{\theta},\theta_0}\|\mathbf{y}-\mathbf{X}\boldsymbol{\Theta}\|^2+\frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$
  Closed form solution: $\hat{\boldsymbol{\Theta}}=\left(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$
  We want to find the best λ (perhaps try a few different λ and check performance on validation data)

Non-linear Rules

- Given $\mathcal{D}\coloneqq\{(x_t,y_t)\}_{t=1}^n$, we can map it to some $\mathcal{D}'\coloneqq\{(\boldsymbol{\phi}(x_t),y_t)\}_{t=1}^n$, and map back the found $\boldsymbol{\theta}$ and θ_0
  - for example, if we want a quadratic expression, we can let $\boldsymbol{\phi}(x)=[x,x^2]^T$; if we want a circle, we can write an equation of a circle
  - if $d>1$, we can also consider cross terms, e.g. x_1x_2 terms
  - if there are too many features, it might be prone to overfitting

Kernel methods

- We want the new dataset to be $k(\mathbf{x}_i,\mathbf{x}_j)\coloneqq\langle\boldsymbol{\phi}(\mathbf{x}_i),\boldsymbol{\phi}(\mathbf{x}_j)\rangle$, where
  - $\boldsymbol{\phi}$ is a "feature map"
  - we compute it without explicitly calculating $\boldsymbol{\phi}(\mathbf{x}_i)$
  - it works nice because inner products capture geometry
  - this trick works only when the rule becomes linear on the new dataset
  - note: the fresh $\mathbf{x}$ whose y is being predicted usually has its kernel with existing data points calculated too, and goes into the prediction function
- **E.g.**: If $\boldsymbol{\phi}(x)=(1,\sqrt{3}x,\sqrt{3}x^2,x^3)$, then we have kernel $\langle\boldsymbol{\phi}(x),\boldsymbol{\phi}(x')\rangle=(1+xx')^3$
  For degree p polynomial and d dimensions, we have the polynomial kernel $k(\mathbf{x},\mathbf{x}')=(1+\langle\mathbf{x},\mathbf{x}'\rangle)^p$
- **E.g.**: String kernel, let $k(\mathbf{x},\mathbf{x}')$ be the number of words appearing in both strings
- Intuitively (non-rigorously), $k(\mathbf{x},\mathbf{x}')$ should be a kind of measure of similarity
- **Def**: $k(\mathbf{x},\mathbf{x}')$ is a kernel: $k(\mathbf{x},\mathbf{x}')=\langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{x}')\rangle$ for some $\boldsymbol{\phi}$ (possibly infinite-dimensional)
- **Positive semidefinite**: $k:\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$ is positive semidefinite: for any $\mathbf{x}$, $\mathbf{x}'$, $k(\mathbf{x},\mathbf{x}')=k(\mathbf{x}',\mathbf{x})$, and for any $\mathbf{x}_1,\dots,\mathbf{x}_m$ (for any m), the Gram matrix
  $$\mathbf{K}\coloneqq\begin{bmatrix}k(\mathbf{x}_1,\mathbf{x}_1)&\cdots&k(\mathbf{x}_1,\mathbf{x}_m)\\\vdots&\ddots&\vdots\\k(\mathbf{x}_m,\mathbf{x}_1)&\cdots&k(\mathbf{x}_m,\mathbf{x}_m)\end{bmatrix}\text{ is positive semidefinite}$$
  (note: positive semidefinite matrix $\implies$ symmetric matrix)
- **Positive semidefinite matrix**: (two equiv. characterisations):
  - $\forall\mathbf{z},\mathbf{z}^T\mathbf{K}\mathbf{z}\geq0$
  - all upper-left square submatrices have nonnegative determinant, or
  - all eigenvalues are nonnegative
- **Thm** k is a kernel $\iff$ k is positive semidefinite
  (useful for proving if something is not a valid kernel)
  Proof for "$\implies$": let $\mathbf{K}=\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ where $\boldsymbol{\Phi}\coloneqq[\boldsymbol{\phi}(\mathbf{x}_1),\dots,\boldsymbol{\phi}(\mathbf{x}_n)]$, and observe that $\mathbf{K}$ is the Gram matrix; then observe that $\forall\mathbf{z},\mathbf{z}^T\mathbf{K}\mathbf{z}=\mathbf{z}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\mathbf{z}=(\boldsymbol{\Phi}\mathbf{z})^T(\boldsymbol{\Phi}\mathbf{z})=\|\boldsymbol{\Phi}\mathbf{z}\|^2\geq0$ (so $\mathbf{K}$ is positive semidefinite)
- **To show that $k(\mathbf{x},\mathbf{x}')$ is a kernel**: (any one)
  - explicitly find $\boldsymbol{\phi}$
  - use kernel closure properties below
  - (rarely, and hard) show that k is positive-semidefinite
- **To show that $k(\mathbf{x},\mathbf{x}')$ is not a kernel**: (any one)
  - k is not symmetric
  - $k(\mathbf{x},\mathbf{x})<0$ for some $\mathbf{x}$
  - Find some $\mathbf{x},\mathbf{x}'$ such that $\det\left(\begin{bmatrix}k(\mathbf{x},\mathbf{x})&k(\mathbf{x},\mathbf{x}')\\k(\mathbf{x}',\mathbf{x})&k(\mathbf{x}',\mathbf{x}')\end{bmatrix}\right)<0$
- **Deriving more kernels**: If k_1,k_2 are kernels, then:
  - $\forall$ functions f (real-valued), $f(\mathbf{x})k_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}')$ is a kernel
  - $k_1(\mathbf{x},\mathbf{x}')+k_2(\mathbf{x},\mathbf{x}')$ is a kernel
  - $k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}')$ is a kernel
  (proof of first: let $\boldsymbol{\phi}(\mathbf{x})=\boldsymbol{\phi}_1(\mathbf{x})f(\mathbf{x})$ and show that $\langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{x}')\rangle=f(\mathbf{x})k_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}')$)
  (proof of second: let $\boldsymbol{\phi}(\mathbf{x})=\begin{bmatrix}\boldsymbol{\phi}_1(\mathbf{x})\\\boldsymbol{\phi}_2(\mathbf{x})\end{bmatrix}$ and show that $\langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{x}')\rangle=k_1(\mathbf{x},\mathbf{x}')+k_2(\mathbf{x},\mathbf{x}')$)
  (proof of third: let $\boldsymbol{\phi}(\mathbf{x})$ be a vector containing entries $\phi_{ij}(\mathbf{x})=\phi_i^{(1)}(\mathbf{x})\phi_j^{(2)}(\mathbf{x})$ for each i, j, and show that $\langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{x}')\rangle=k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}')$)
- **RBF kernel (aka. Gaussian kernel)**: $k(\mathbf{x},\mathbf{x}')=\exp(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2)$
  - measures distance between points; larger (and tends to 1) as $\mathbf{x}'\to\mathbf{x}$
  - has infinite number of features ($\boldsymbol{\phi}(\mathbf{x})$), and it is not easy to determine
- **RBF kernel (aka. Gaussian kernel) with length scale**: $k(\mathbf{x},\mathbf{x}')=\exp(-\frac{1}{2\ell}\|\mathbf{x}-\mathbf{x}'\|^2)$
  - ℓ = length scale, increasing ℓ means that points don't need to be too close in order to be considered similar
- **Some "kernelizable" learning algorithms**:
  - k-nearest neighbours: Want to make a prediction for $\mathbf{x}$ (where $\hat{y}\in\{-1,1\}$). Given $\mathbf{x}$, we find the k nearest points, and predict $\hat{y}$ by majority rule. This is kernelizable because $\|\mathbf{x}-\mathbf{x}_t\|$ can be expressed as inner products.
  - Linear (ridge) regression: After solving $\boldsymbol{\Theta}$, want to predict $\langle\boldsymbol{\Theta},\mathbf{x}\rangle$ given $\mathbf{x}$; observe that $\left(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}_d\right)^{-1}\mathbf{X}^T=\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T+\lambda\mathbf{I}_n\right)^{-1}$, so $\hat{\boldsymbol{\Theta}}=\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T+\lambda\mathbf{I}_n\right)^{-1}\mathbf{y}$, so $\hat{y}=\mathbf{x}^T\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T+\lambda\mathbf{I}_n\right)^{-1}\mathbf{y}$ (for no-offset case). Observe that $\mathbf{X}\mathbf{X}^T$ is just the Gram matrix with standard inner products (i.e. a matrix of inner products), and $\mathbf{x}^T\mathbf{X}^T$ is a vector of inner products, so we can replace them with inner products and this prediction rule (for $\hat{y}$) would be kernelisable, i.e. $\hat{y}=\mathbf{k}(\mathbf{x})(\mathbf{K}+\lambda\mathbf{I}_n)^{-1}\mathbf{y}$ where $\mathbf{K}$ is the Gram matrix and $\mathbf{k}(\mathbf{x})=\begin{bmatrix}k(\mathbf{x},\mathbf{x}_1)\\\vdots\\k(\mathbf{x},\mathbf{x}_n)\end{bmatrix}$

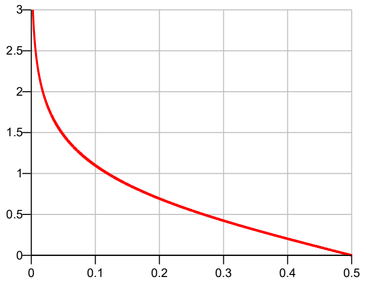
Convex Optimisation (for Kernel SVM)

- $\mathcal{D}\subseteq\mathbb{R}^d$ is a **convex set**: $\forall\mathbf{x},\mathbf{x}'\in\mathcal{D},\forall\lambda\in[0,1],\lambda\mathbf{x}+(1-\lambda)\mathbf{x}'\in\mathcal{D}$
- $f:D\to\mathbb{R}$ (where $D\subseteq\mathbb{R}^d$ is convex) is a **convex function**: $\forall\mathbf{x},\mathbf{x}'\in\mathcal{D},\forall\lambda\in[0,1],f(\lambda\mathbf{x}+(1-\lambda)\mathbf{x}')\leq\lambda f(\mathbf{x})+(1-\lambda)f(\mathbf{x}')$
  (for **concave function**, change "$\leq$" to "$\geq$")
- Equiv. def if f differentiable: $f(\mathbf{x}')\geq f(\mathbf{x})+\nabla f(\mathbf{x})^T(\mathbf{x}'-\mathbf{x})$
  (i.e. the second point lies above the tangent from the first point)
- Equiv. def if f twice differentiable: $\nabla^2f(\mathbf{x})\geq\mathbf{0}$ (i.e. is positive semidefinite) (for 1D case, $f''(x)\geq0$)
- **Properties of convex functions**:
  - f_1,f_2 convex $\implies$ $\forall\alpha_1,\alpha_2\geq0,\alpha_1f_1(\mathbf{x})+\alpha_2f_2(\mathbf{x})$ convex
  - $f_1,\dots f_n$ convex $\implies$ $\max_if_i(\mathbf{x})$ convex
  - (Jensen's): $f(\mathbb{E}[\mathbf{X}])\leq\mathbb{E}[f(\mathbf{X})]$
- **Convex optimisation**: minimise some convex function subject to some constraints:
  Minimise (over $\mathbf{x}$)) $f_0(\mathbf{x})$ (where f_0 convex) such that $f_0(\mathbf{x})$ subject to $f_i(\mathbf{x})\leq0$ (where f_i convex) and $h_i(\mathbf{x})=0$ (where h_i affine (i.e. linear))
- **Lagrangian**: $L(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\nu})=f_0(\mathbf{x})+\sum_i\lambda_if_i(\mathbf{x})+\sum_i\nu_ih_i(\mathbf{x})$
  ($\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are called "dual variables"; can think of it like $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are penalties that we want to minimise)
  $\underbrace{\min_{\mathbf{x}}\max_{\boldsymbol{\lambda},\boldsymbol{\nu},\boldsymbol{\lambda}\geq0}L(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\nu})}_{\text{original problem}}=\underbrace{\max_{\boldsymbol{\lambda},\boldsymbol{\nu},\boldsymbol{\lambda}\geq0}\min_{\mathbf{x}}L(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\nu})}_{\text{equivalent dual problem}}$ due to nice convexity properties satisfying Minimax theorem; the RHS is often easier to solve
  (note: $\min_{\mathbf{x}}L(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\nu})$ is usually called $g(\boldsymbol{\lambda},\boldsymbol{\nu})$)
  (if the min point of $\mathbf{x}$ has $\lambda_i=0$, then f_i is inactive)
- **Karush-Kuhn-Tucker (KKT) Conditions** (that the minimum point $(\mathbf{x}^*,\boldsymbol{\lambda}^*,\boldsymbol{\nu}^*)$ must satisfy):
  - Primal feasibility: $f_i(\mathbf{x}^*)\leq0\ \forall i=1,\dots,m_{\text{ineq}}$ and $h_i(\mathbf{x}^*)=0\ \forall i=1,\dots,m_{\text{eq}}$
  - Dual feasibility: $\lambda_i^*\geq0\ \forall i=1,\dots,m_{\text{ineq}}$
  - Complementary slackness: $\lambda_i^*f_i(\mathbf{x}^*)=0\ \forall i=1,\dots,m_{\text{ineq}}$
    (intuitively, either the min. pt. is on the boundary ($f_i(\mathbf{x})=0$) or the constraint is inactive ($\lambda_i^*=0$))
  - Vanishing gradient: $\nabla f_0(\mathbf{x}^*)+\sum_{i=1}^{m_{\text{ineq}}}\lambda_i^*\nabla f_i(\mathbf{x}^*)+\sum_{i=1}^{m_{\text{eq}}}\nu_i^*\nabla h_i(\mathbf{x}^*)=\mathbf{0}$
  In the general case this is necessary but not sufficient, but in the convex case it is both necessary and sufficient
- **To solve a convex optimisation problem**:
  - write out the Lagrangian
  - solve the dual problem by finding $g(\boldsymbol{\lambda},\boldsymbol{\nu})$, and then taking the maximum over $\boldsymbol{\lambda},\boldsymbol{\nu}$
- **E.g. Hard margin SVM**: minimise (over $\boldsymbol{\theta}$, θ_0) $\frac{1}{2}\|\boldsymbol{\theta}\|^2$ subject to $\forall t$, $y_t\left(\boldsymbol{\theta}^T\mathbf{x}_t+\theta_0\right)\geq1$
  - $L(\boldsymbol{\theta},\theta_0,\boldsymbol{\lambda})=\frac{1}{2}\|\boldsymbol{\theta}\|^2+\sum_{t=1}^n\lambda_t(1-y_t(\boldsymbol{\theta}^T\mathbf{x}_t+\theta_0))$
  - $\frac{\partial L}{\partial\theta_0}=\sum_{t=1}^n(-\lambda_ty_t)=0$ (equate to zero because we want the min point)
  - $\frac{\partial L}{\partial\boldsymbol{\theta}}=\boldsymbol{\theta}-\sum_{t=1}^n(\lambda_ty_t\mathbf{x}_t)=0$ (equate to zero because we want the min point), so $\boldsymbol{\theta}^*=\sum_{t=1}^n(\lambda_ty_t\mathbf{x}_t)$
  - observe that at the min point,
  $\sum_{t=1}^n\lambda_t(1-y_t(\boldsymbol{\theta}^T\mathbf{x}_t+\theta_0))$
  $=\sum_{t=1}^n\lambda_t-\sum_{t=1}^n\lambda_ty_t\boldsymbol{\theta}^T\mathbf{x}_t-\sum_{t=1}^n\lambda_ty_t\theta_0$
  $=\sum_{t=1}^n\lambda_t-\sum_{t=1}^n\lambda_ty_t\left(\sum_{t=1}^n(\lambda_ty_t\mathbf{x}_t)\right)^T\mathbf{x}_t-\theta_0\underbrace{\sum_{t=1}^n\lambda_ty_t}_{=0}$
  $=\sum_{t=1}^n\lambda_t-\left(\sum_{s=1}^n(\lambda_sy_s\mathbf{x}_s)\right)^T\sum_{t=1}^n\lambda_ty_t\mathbf{x}_t$
  $=\sum_{t=1}^n\lambda_t-\|\boldsymbol{\theta}^*\|^2$
  - so at the min point, $L(\boldsymbol{\theta},\theta_0,\boldsymbol{\lambda})=\sum_{t=1}^n\lambda_t-\frac{1}{2}\|\boldsymbol{\theta}\|^2=\sum_{t=1}^n\lambda_t-\frac{1}{2}\sum_{s=1}^n\sum_{t=1}^n\lambda_s\lambda_ty_sy_t\mathbf{x}_s^T\mathbf{x}_t$
  - more formally,
  $g(\boldsymbol{\lambda})=\begin{cases}\sum_{t=1}^n\lambda_t-\frac{1}{2}\sum_{s=1}^n\sum_{t=1}^n\lambda_s\lambda_ty_sy_t\mathbf{x}_s^T\mathbf{x}_t&\text{if}\sum_{t=1}^n\lambda_ty_t=0\\-\infty&\text{otherwise}\end{cases}$
  (since if $\sum_{t=1}^n\lambda_ty_t\neq0$ we can choose θ_0 to be very big or very small, in order to make $\theta_0\sum_{t=1}^n\lambda_ty_t\to-\infty$)
  - to find the maximum of $g(\boldsymbol{\lambda})$ (over $\boldsymbol{\lambda}\geq0$), solve maximise$_\alpha\sum_{t=1}^n\alpha_t-\frac{1}{2}\sum_{s=1}^n\sum_{t=1}^n\alpha_s\alpha_ty_sy_t\underbrace{\mathbf{x}_s^T\mathbf{x}_t}_{\text{kernelisable}}$ subject to $\alpha_t\geq0$ and $\sum_t\alpha_ty_t=0$
  - then the classifier is $\boldsymbol{\theta}=\sum_{t=1}^n\alpha_ty_t\mathbf{x}_t$, and $\theta_0=\frac{1}{y_t}-\boldsymbol{\theta}^T\mathbf{x}_t$ for any point $(\mathbf{x}_t,y_t)$ where $\alpha_t>0$ (i.e. $y_t(\boldsymbol{\theta}^T\mathbf{x}_t+\theta_0)=1$, i.e. $\mathbf{x}_t$ is a support vector)
- **Computation of Primal vs Dual SVM**:
  Primal: Solving an optimisation problem with d variables and n (or $2n$) constraints (better for $n\gg d$)
  Dual: Form $\frac{n^2}{2}$ kernel values (each of the $\mathbf{x}_s$, $\mathbf{x}_t$ pairs), then solve an optimisation problem with n variables and $n+1$ constraints (better for $d\gg n$ if can compute $k(\mathbf{x}_s,\mathbf{x}_t)$ efficiently)

Boosting

- **Decision stump**: A linear classifier where the classifier must be either horizontal or vertical (with offset) (no diagonal lines)
  $h(\mathbf{x};\boldsymbol{\theta})\coloneqq\operatorname{sgn}(s(x_k-\theta_0))$ where $\boldsymbol{\theta}=(s,k,\theta_0)$, where $s\in\{-1,1\}$, $k\in\{1,\dots,d\}$, $\theta_0\in\mathbb{R}$
  (intuitively, s is the direction, k is the dimension index, θ_0 is the threshold/offset)
  - we want to determine good $\boldsymbol{\theta}$
- **Combined classifier**: $\hat{y}\coloneqq\operatorname{sgn}\left(\sum_{m=1}^M\alpha_mh(\mathbf{x};\boldsymbol{\theta}_m)\right)$ where $\alpha_m\geq0$ are weights (i.e. the vote multiplier of base learner m)

- we want to determine good $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M$

- **Adaboost algorithm** for learning $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M$ and $\alpha_1, \dots, \alpha_M$
  - Input: $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, number of iterations M
  1. Init weights $w_0(t) := \frac{1}{n}$ for $t = 1, \dots, n$
  2. For $m = 1, \dots, M$:
  a. Choose $h(\cdot; \hat{\boldsymbol{\theta}}_m)$ as $\hat{\boldsymbol{\theta}}_m := \arg\min_{\boldsymbol{\theta}} \sum_{\substack{t=1 \\ y_t \neq h(\mathbf{x}_t; \boldsymbol{\theta})}}^n w_{m-1}(t)$
  (i.e. minimise the weight of all misclassified points)
  (note: $\arg\min_{\boldsymbol{\theta}} \sum_{\substack{t=1 \\ y_t \neq h(\mathbf{x}_t; \boldsymbol{\theta})}}^n w_{m-1}(t) = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^n (-y_t h(\mathbf{x}_t; \boldsymbol{\theta})) w_{m-1}(t)$ since $-y_t h(\mathbf{x}_t; \boldsymbol{\theta}) = 2 \cdot \mathbf{1}\{y_t \neq h(\mathbf{x}_t; \boldsymbol{\theta})\} - 1$)
  b. $\hat{\alpha}_m := \frac{1}{2} \log \frac{1 - \hat{\varepsilon}_m}{\hat{\varepsilon}_m}$ where $\hat{\varepsilon}_m := \sum_{\substack{t=1 \\ y_t \neq h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m)}}^n w_{m-1}(t)$
  (plot of $\hat{\alpha}_m$ against $\hat{\varepsilon}_m$: 

)
  c. Update weights: $w_m(t) := \frac{1}{Z_m} w_{m-1}(t) e^{-y_t h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m) \hat{\alpha}_m}$
  where $Z_m := \sum_{t=1}^n w_{m-1}(t) e^{-y_t h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m) \hat{\alpha}_m}$ is the normalization factor (i.e. we want $\sum_{t=1}^n w_m(t) = 1$) (note: $e^{\hat{\alpha}_m} > 1$ and $e^{-\hat{\alpha}_m} < 1$ since $\hat{\alpha}_m > 0$, so it increases relative weights for misclassified points w.r.t. current decision stump)
  3. The output classifier is $f_M(\mathbf{x}) := \sum_{m=1}^M \hat{\alpha}_m h(\mathbf{x}; \hat{\boldsymbol{\theta}}_m)$

- Note: Adaboost never overfits points for some reason

- **Training error**: fraction of misclassified points: $\frac{1}{n} \sum_{t=1}^n \mathbf{1}\{y_t f_m(\mathbf{x}) \leq 0\}$

- **Theorem**: After M iterations,
  training error $\leq \exp\left(-2 \sum_{m=1}^M \left(\frac{1}{2} - \hat{\varepsilon}_m\right)^2\right)$
  (in particular, if $\hat{\varepsilon}_m \leq \frac{1}{2} - \gamma \ \forall m$, then training error $\leq \exp\left(-2M\gamma^2\right)$; and since training error must be a multiple of $\frac{1}{n}$, if $\exp\left(-2M\gamma^2\right) < \frac{1}{n}$ then training error must be zero (i.e. everything classified correctly))

- **Adaboost proof**:
  1. Bound 0-1 loss with the exponential loss:
  $\frac{1}{n} \sum_{t=1}^n \mathbf{1}\{y_t f_M(\mathbf{x}) \leq 0\} \leq \frac{1}{n} \sum_{t=1}^n \exp(-y_t f_M(\mathbf{x}))$ (proof: it is obvious)
  2. By weight update formula,
  $w_M(t) = \frac{1}{n} \prod_{m=1}^M \frac{\exp(-y_t h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m) \hat{\alpha}_m)}{Z_m} = \frac{1}{n} \frac{\exp(-\sum_{m=1}^M y_t h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m) \hat{\alpha}_m)}{\prod_{m=1}^M Z_m} = \frac{1}{n} \frac{\exp(-y_t f_M(\mathbf{x}))}{\prod_{m=1}^M Z_m}$
  Hence $1 = \sum_{t=1}^n w_M(t) = \frac{1}{n} \frac{\sum_{t=1}^n \exp(-y_t f_M(\mathbf{x}))}{\prod_{m=1}^M Z_m}$, and thus
  $\frac{1}{n} \sum_{t=1}^n \exp(-y_t f_M(\mathbf{x})) = \prod_{m=1}^M Z_m$
  Combining with step 1,
  $\frac{1}{n} \sum_{t=1}^n \mathbf{1}\{y_t f_M(\mathbf{x}) \leq 0\} \leq \prod_{m=1}^M Z_m$
  3. $Z_m = \sum_{t=1}^n w_{m-1}(t) e^{-y_t h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m) \hat{\alpha}_m} = \sum_{t: y_t \neq h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m)} w_{m-1}(t) e^{\hat{\alpha}_m} + \sum_{t: y_t = h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m)} w_{m-1}(t) e^{-\hat{\alpha}_m} = e^{\hat{\alpha}_m} \sum_{t: y_t \neq h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m)} w_{m-1}(t) + e^{-\hat{\alpha}_m} \sum_{t: y_t = h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m)} w_{m-1}(t) = \hat{\varepsilon}_m e^{\hat{\alpha}_m} + (1 - \hat{\varepsilon}_m) e^{-\hat{\alpha}_m}$
  So we want to pick $\hat{\alpha}_m$ to minimise Z_m
  (proof skipped... the optimum $\hat{\alpha}_m = \frac{1}{2} \log \frac{1 - \hat{\varepsilon}_m}{\hat{\varepsilon}_m}$)
  4. Substitute $\hat{\alpha}_m$ to step 3, and get
  $Z_m = \hat{\varepsilon}_m \sqrt{\frac{1 - \hat{\varepsilon}_m}{\hat{\varepsilon}_m}} + (1 - \hat{\varepsilon}_m) \sqrt{\frac{\hat{\varepsilon}_m}{1 - \hat{\varepsilon}_m}} = 2\sqrt{\hat{\varepsilon}_m (1 - \hat{\varepsilon}_m)} = \sqrt{1 - (1 - 2\hat{\varepsilon}_m)^2} = \exp\left(\frac{1}{2} \log\left(1 - (1 - 2\hat{\varepsilon}_m)^2\right)\right) \leq \exp\left(-\frac{1}{2} (1 - 2\hat{\varepsilon}_m)^2\right) = \exp\left(-2 \left(\frac{1}{2} - \hat{\varepsilon}_m\right)^2\right)$
  (note: to remove log, use formula $\log(1 + a) \leq a \ \forall a \in \mathbb{R}$)
  5. Substitute into step 2 end, we get
  $\frac{1}{n} \sum_{t=1}^n \mathbf{1}\{y_t f_M(\mathbf{x}) \leq 0\} \leq \exp\left(-2 \sum_{m=1}^M \left(\frac{1}{2} - \hat{\varepsilon}_m\right)^2\right)$

- **Claim**: $\sum_{t: y_t \neq h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m)} w_m(t) = \frac{1}{2}$ (hence the same $\hat{\boldsymbol{\theta}}$ will not be chosen twice in a row)
  Proof: LHS $= \frac{1}{2} \iff \sum_{t=1}^n (-y_t h(\mathbf{x}_t; \boldsymbol{\theta})) w_m(t) = 0$, and we have $\sum_{t=1}^n (-y_t h(\mathbf{x}_t; \boldsymbol{\theta})) w_m(t) = \sum_{t: y_t = h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m)} w_m(t) + \sum_{t: y_t \neq h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m)} (-w_m(t)) = \sum_{t: y_t = h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m)} \frac{1}{Z_m} w_{m-1}(t) e^{-\hat{\alpha}_m} - \sum_{t: y_t \neq h(\mathbf{x}_t; \hat{\boldsymbol{\theta}}_m)} \frac{1}{Z_m} w_{m-1}(t) e^{\hat{\alpha}_m} = \frac{1}{Z_m} \left(e^{-\hat{\alpha}_m} (1 - \hat{\varepsilon}_m) - e^{\hat{\alpha}_m} \hat{\varepsilon}_m\right) = 0$ (last step is by definition of $\hat{\alpha}_m$ being chosen to minimise the expression obtained in step 3 of Adaboost proof)

Theory

Concentration

- **Concentration**: General idea: to show how well things concentrate around the mean:
  $\mathbb{P}[|Y - m| > t] \leq \text{TailBound}(t)$

- Consider $Y_n = \frac{1}{n} \sum_{i=1}^n X_i$ where $\mathbb{E}[X_i] = \mu$ and $\text{Var}[X_i] = \sigma^2$ and X_i are i.i.d.
  Law of large numbers: $\mathbb{P}[|Y_n - \mu| > \varepsilon] \to 0$ as $n \to \infty$ for any $\varepsilon > 0$
  Central limit theorem: $\mathbb{P}[|Y_n - \mu| > \frac{\alpha}{\sqrt{n}}] \to 2\Phi(-\frac{\alpha}{\sigma})$ as $n \to \infty$ where Φ is the standard normal c.d.f. (inaccurate for very small probabilities)
  Large deviations theory (important):
  $\mathbb{P}[|Y_n - \mu| > \varepsilon] \leq e^{-n \cdot \psi(\varepsilon)}$ (this is true for any ε and any n, not just for large n)

- **Basic inequalities**:
  Markov's ineq.: If Z is a nonnegative rand. var., then $\mathbb{P}[Z \geq t] \leq \frac{\mathbb{E}[Z]}{t}$
  Markov's ineq. for functions: If Z is a rand. var. and ϕ is a nonnegative increasing function, then
  $\mathbb{P}[Z \geq t] \leq \mathbb{P}[\phi(Z) \geq \phi(t)] \leq \frac{\mathbb{E}[\phi(Z)]}{\phi(t)}$
  Chebyshev's ineq.: If Z is a rand. var., then
  $\mathbb{P}[|Z - \mathbb{E}[Z]| \geq t] \leq \frac{\text{Var}[Z]}{t^2}$ (proof by letting $\phi(t) = t^2$ and replacing Z by $|Z - \mathbb{E}[Z]|$)
  Chernoff bound: If Z is a rand. var. and $\lambda \geq 0$, then
  $\mathbb{P}[Z \geq t] \leq e^{-\lambda t} \mathbb{E}[e^{\lambda Z}]$ (proof by letting $\phi(t) = e^{\lambda t}$)

- **Sum of independent random variables**:
  $Z = X_1 + \cdots + X_n$ and $Y_n = \frac{1}{n} Z$ where X_i are i.i.d.:
  Chebyshev's ineq.: $\mathbb{P}[|Y_n - \mathbb{E}[Y_n]| \geq \varepsilon] \leq \frac{\text{Var}[X]}{n\varepsilon^2}$
  Chernoff bound: $\mathbb{P}[Z \geq n\varepsilon] \leq \exp(-n \cdot \psi_X^*(\varepsilon))$ where $\psi_X^*(t) = \max_\lambda (\lambda t - \psi_X(\lambda))$ and $\psi_X(\lambda) = \text{Log}\, \mathbb{E}[e^{\lambda X}]$
  (e.g. for Gaussian ($X \sim N(0, \sigma^2)$), $\psi_X(\lambda) = \frac{\lambda^2 \sigma^2}{2}$, so $\psi_X^*(t) = \frac{t^2}{2\sigma^2}$, so $\mathbb{P}[Z \geq n\varepsilon] \leq \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right)$, so
  $\mathbb{P}[|Z| \geq n\varepsilon] \leq 2 \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right)$)
  (note that for σ^2-sub-Gaussian, we instead assume that $\psi_X(\lambda) \leq \frac{\lambda^2 \sigma^2}{2}$, and the same result holds)

- **Hoeffding's ineq.**: $Z = X_1 + \cdots + X_n$ where X_i are i.i.d. and $X_i \in [a_i, b_i]$:
  $\mathbb{P}\left[\frac{1}{n} |Z - \mathbb{E}[Z]| \geq \varepsilon\right] \leq 2 \exp\left(\frac{-2n\varepsilon^2}{\frac{1}{n} \sum_{i=1}^n (b_i - a_i)^2}\right)$
  If $a_i = a$ and $b_i = b$ for all i then:
  $\mathbb{P}\left[\frac{1}{n} |Z - \mathbb{E}[Z]| \geq \varepsilon\right] \leq 2 \exp\left(\frac{-2n\varepsilon^2}{(b-a)^2}\right)$
  If $a_i = 0$ and $b_i = 1$ for all i then:
  $\mathbb{P}\left[\frac{1}{n} |Z - \mathbb{E}[Z]| \geq \varepsilon\right] \leq 2 \exp\left(-2n\varepsilon^2\right)$

Statistical learning theory

- **Underfitting**: High training error (and hence high test error)
  Overfitting: Low training error but high test error

- **Setup**:
  - Data drawn from distribution $P_{\mathbf{X}Y}$ (unknown), i.e. $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, where each $(\mathbf{x}_t, y_t) \sim P_{\mathbf{X}Y}$ and i.i.d.
  - Loss function $\ell(y, \hat{y})$ where $\hat{y} = f(x)$ (where f is our classifier) (ℓ could be any loss function)
  - Training error (empirical risk): $R_n(f) = \frac{1}{n} \sum_{t=1}^n \ell(y_t, f(\mathbf{x}_t))$
  - Test error (true risk): $R(f) = \mathbb{E}[\ell(y, f(\mathbf{x}))]$ where $(x, y) \sim P_{\mathbf{X}Y}$
  - $\underbrace{R(f)}_{\text{Test error}} = \underbrace{R_n f}_{\text{Training error}} + \underbrace{(R(f) - R_n f)}_{\text{Generalisation error}}$
  (Generalisation error is large if we overfit)
  - Consider algorithm that outputs $f \in \mathcal{F}$ (if $\mathcal{F}$ is too small then we underfit; if $\mathcal{F}$ is too large then we overfit)

- **Task**: Let $f_{\text{erm}} = \arg\min_{f \in \mathcal{F}} R_n(f)$, want to find out if f_{erm} has small $R(f)$ too? Will show that $R(f_{\text{erm}}) \leq R(f^*) + \varepsilon$ with probability $\geq 1 - \delta$ provided that $n \geq \overline{n}(\varepsilon, \delta)$ (where $f^* = \arg\min_{f \in \mathcal{F}} R(f)$, and $\overline{n}$ is known as the sample complexity) (called the "probably approximately correct" (PAC) guarantee) (the analysis is true even for the worst case $P_{\mathbf{X}Y}$, so for good distributions, it may be possible to use less data points)
  $\mathcal{F}$ is PAC-learnable iff this is attainable for all $\varepsilon > 0$, $\delta > 0$ with $\overline{n} < \infty$ (regardless of $P_{\mathbf{X}Y}$) (when $|\mathcal{F}|$ is small enough, this is usually possible)
  Realisable: $y_i = f(\mathbf{x}_i)$ for all i, for some $f \in \mathcal{F}$
  Agnostic: Just compare with the (ideal) $f^* \in \mathcal{F}$ (we usually use this)

- **PAC guarantee for finite $\mathcal{F}$**: If $|\mathcal{F}| < \infty$ and $\ell(y, \hat{y}) \in [0, 1]$, then $\mathcal{F}$ is PAC-learnable (i.e. $R(f_{\text{erm}}) \leq R(f^*) + \varepsilon$ with probability $\geq 1 - \delta$) with $\overline{n}_{\mathcal{F}}(\varepsilon, \delta) = \frac{2}{\varepsilon^2} \log \frac{2|\mathcal{F}|}{\delta}$
  (equivalently, $R(f_{\text{erm}}) - R(f^*) \leq \varepsilon \leq \sqrt{\frac{2}{n} \log \frac{2|\mathcal{F}|}{\delta}}$)

- **Proof of PAC guarantee for finite $\mathcal{F}$**:
  1. Fix $f \in \mathcal{F}$. Let $z_i = \ell(y_i, f(\mathbf{x}_i)) \ \forall i$. Then since $(\mathbf{x}_i, y_i)$ are i.i.d., z_i is also i.i.d.. Since $z_i \in [0, 1]$, and $\mathbb{E}[z_i] = \mathbb{E}[\ell(y_i, f(\mathbf{x}_i))] = R(f)$, we apply Hoeffding's ineq to get $\mathbb{P}\left[\left|\frac{1}{n} \sum_{i=1}^n z_i - R(f)\right| > \varepsilon_0\right] \leq 2 \exp\left(-2n\varepsilon_0^2\right)$. Then observe that $\frac{1}{n} \sum_{i=1}^n z_i = \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) = R_n(f)$, so $\mathbb{P}\left[|R_n(f) - R(f)| > \varepsilon_0\right] \leq 2 \exp\left(-2n\varepsilon_0^2\right)$.
  2. Take the union bound:
  $\mathbb{P}\left[\bigcup_{f \in \mathcal{F}} \{|R_n(f) - R(f)| > \varepsilon_0\}\right] \leq 2 |\mathcal{F}| \exp\left(-2n\varepsilon_0^2\right)$
  3. So we pick $\delta = 2 |\mathcal{F}| \exp\left(-2n\varepsilon_0^2\right)$, so $n = \frac{1}{2\varepsilon_0^2} \log \frac{2|\mathcal{F}|}{\delta}$.
  4. Suppose $|R_n(f) - R(f)| \leq \varepsilon_0 \ \forall f \in \mathcal{F}$ (holds with probability $\geq 1 - \delta$), then: $R(f_{\text{erm}}) - R(f^*) = \underbrace{R(f_{\text{erm}}) - R_n(f_{\text{erm}})}_{\leq \varepsilon_0} + \underbrace{R_n(f_{\text{erm}}) - R_n(f^*)}_{\leq 0} + \underbrace{R_n(f^*) - R(f^*)}_{\leq \varepsilon_0} \leq 2\varepsilon_0$. Hence we let $\varepsilon := 2\varepsilon_0$, so $n = \frac{2}{2\varepsilon^2} \log \frac{2|\mathcal{F}|}{\delta}$.

- **Shattering**: A set of points $\mathbf{x}_1, \dots, \mathbf{x}_k$ is said to be shattered by $\mathcal{F}$ if $|\{(f(\mathbf{x}_1), \dots, f(\mathbf{x}_k)) \mid f \in \mathcal{F}\}| = 2^k$

- **VC Dimension**: $d_{\text{VC}} = d_{\text{VC}}(\mathcal{F}) =$ largest k such that $\exists \mathbf{x}_1, \dots, \mathbf{x}_k$ that $\mathcal{F}$ shatters (i.e. largest k such that $\exists \mathbf{x}_1, \dots, \mathbf{x}_k$ for which all combinations of labels (i.e. $f(\mathbf{x}_1), \dots, f(\mathbf{x}_k)$) can be produced by classifiers in $\mathcal{F}$)
  (to determine the VC dimension, must show that k works but $k + 1$ does not work)

- **Corollary for Finite $\mathcal{F}$**: If $\mathcal{F}$ is finite, $d_{\text{VC}} \leq \log_2 |\mathcal{F}|$

- **Sauer's lemma**: Effective $\#f$ in $\mathcal{F} \leq \sum_{i=0}^{d_{\text{VC}}} \binom{n}{i}$
  Slightly weaker version: Effective $\#f$ in $\mathcal{F} \leq (n + 1)^{d_{\text{VC}}}$

- **PAC guarantee for infinite $\mathcal{F}$**: Assuming 0-1 loss, $\mathcal{F}$ is PAC-learnable with $\overline{n}_{\mathcal{F}}(\varepsilon, \delta) = C \cdot \frac{d_{\text{VC}}(\mathcal{F}) + \log \frac{1}{\delta}}{\varepsilon^2}$

- **Converse guarantee**: If $d_{\text{VC}} = \infty$, then $\mathcal{F}$ is not PAC-learnable (though there may be certain choices of $P_{\mathbf{X}Y}$ where learning might still work)

- **Examples**
  - Linear classifier (without offset): $d_{\text{VC}} = d$ (the dimension of the space)
  - Linear classifier (with offset): $d_{\text{VC}} = d + 1$

Unsupervised Learning

- **Training data**: $\mathcal{D} = \{\mathbf{x}_t\}_{t=1}^n$

- **K-means clustering**:
  Goal: Partition $\mathcal{D}$ into clusters $\mathcal{D}_1, \dots, \mathcal{D}_k$ such that the associated cluster centres $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K \in \mathbb{R}^d$ minimise $J\left(\{\mathcal{D}_j\}_{j=1}^K, \{\boldsymbol{\mu}_j\}_{j=1}^K\right) = \sum_{j=1}^K \sum_{\mathbf{x} \in \mathcal{D}_j} \left\|\mathbf{x} - \boldsymbol{\mu}_j\right\|^2$ (i.e. minimise the sum of squared distance from the respective cluster centres)

- **K-means algorithm**:
  1. Minimise $J\left(\{\mathcal{D}_j\}_{j=1}^K, \{\boldsymbol{\mu}_j\}_{j=1}^K\right)$ w.r.t. $\{\mathcal{D}_j\}_{j=1}^K$ for fixed $\{\boldsymbol{\mu}_j\}_{j=1}^K$ (i.e. associate each data point with the cluster centre closest to it)
  2. Minimise $J\left(\{\mathcal{D}_j\}_{j=1}^K, \{\boldsymbol{\mu}_j\}_{j=1}^K\right)$ w.r.t. $\{\boldsymbol{\mu}_j\}_{j=1}^K$ for fixed $\{\mathcal{D}_j\}_{j=1}^K$ (i.e. set each cluster centre to the mean of the data points associated with it, i.e. $\boldsymbol{\mu}_j = \frac{1}{|\mathcal{D}_j|} \sum_{\mathbf{x} \in \mathcal{D}_j} \mathbf{x}$)
  3. Repeat until no change
  Notes:
  - initially, choose $\{\boldsymbol{\mu}_j\}_{j=1}^K$ at random
  - K-means finds a local minimum for J

- **Distribution learning**:
  Goal: Estimate a distribution $\hat{p}(\mathbf{x})$ that models the data well (either p.m.f. (discrete) or p.d.f. (continuous))
  - assume $p(\mathbf{x})$ is determined by $\boldsymbol{\theta}$, i.e. $p(\mathbf{x}; \boldsymbol{\theta})$
  - to find $\hat{\boldsymbol{\theta}}$, we use maximum likelihood, i.e.
  $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \prod_{t=1}^n p(\mathbf{x}_t; \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{t=1}^n \log p(\mathbf{x}_t; \boldsymbol{\theta})$