Statistical Machine Learning
Bayesian Linear Classification

Luca Bortolussi

Data Science and Scientific Computing

1 Linear Classifiers

- Data: $\mathbf{x_i}, t_i$. Output are discrete, either binary or multiclass (K classes), and are also denoted by y_i. Classes are denoted by $C_1, \ldots, C_K$.
- *Discriminant function*: we construct a function $f(\mathbf{x}) \in \{1, \ldots, K\}$ associating with each input a class.
- *Generative approach*: We consider a prior over classes, $p(C_k)$, and the class-conditional densities $p(\mathbf{x}|C_k)$, from a parametric family. We learn class-conditional densities from data, and then compute the class posterior.

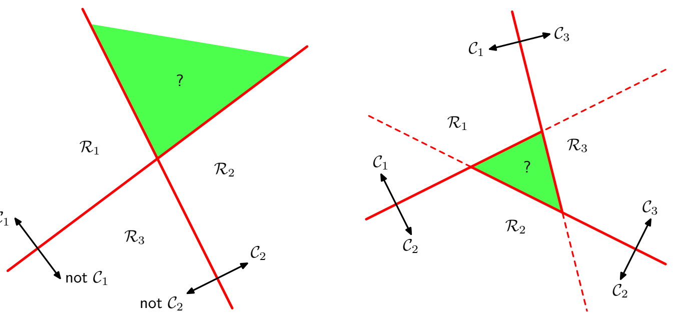
$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})}$$

- *Discriminative approach*: we learn directly a model for the class posteriori $p(C_k|\mathbf{x})$, typically as $p(C_k|\mathbf{x}) = f(\mathbf{w}\boldsymbol{\phi}(\mathbf{x}))$.
  f is called an *activation function* (and f^{-1} a *link function*).

1.1 Encoding of the output and Multi-class strategies

- For a binary classification problem, usually we choose $t_n \in \{0, 1\}$. The interpretation is that of a "probability" to belong to class C_1.
- In some circumstances (perceptron, SVM), we will prefer the encoding $t_n \in \{-1, 1\}$.
- For a multiclass problem, we usually stick to a boolean encoding: $\mathbf{t_n} = (t_{n,1}, \ldots, t_{n,K})$, with $t_{n,j} \in \{0, 1\}$, and t_n is in class k if and only if $t_{n,k} = 1$ and $t_{n,j} = 0$, for $j \neq k$.
- Assume we have a binary classifier. We can train K classifiers, *one-versus-the-rest* strategy, class C_k versus all other points (unbalanced).
- Alternatively, there is the *one-versus-one* classifier, trains $K(K-1)/2$ for each pair of classes, decode by majority voting. Both are ambiguous.

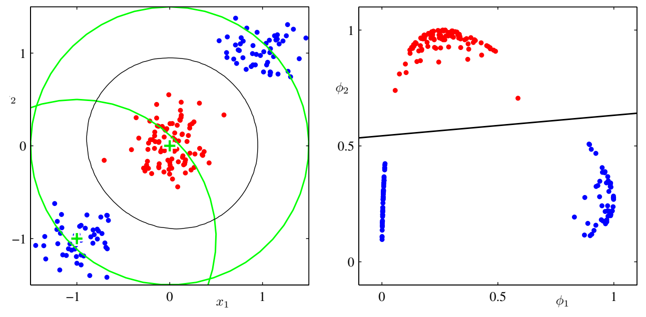
- One can train K linear discriminants $y_k(\mathbf{x}) = \mathbf{w_k}^T\mathbf{x} + b_k$ and decode to j such that $y_j(\mathbf{x}) > y_i(\mathbf{x})$ for each $i \neq j$.



2 Logistic Regression

2.1 Logit and Probit

- We model directly the conditional class probabilities $p(C_1|\mathbf{x}) = f(\mathbf{w}^T\phi(\mathbf{x}))$, after a (nonlinear) mapping of the features $\phi(\mathbf{x}) = \phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})$.
- Common choices for f are the logistic or logit function $\sigma(a) = \frac{1}{1+e^{-a}}$ and the probit function $\psi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta|0,1)d\theta$.
- We will focus on logistic regression.
- The non-linear embedding is an important step



2.2 Logistic regression

- We assume $p(C_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T\phi)$ where $\phi = \phi(\mathbf{x})$ and $\phi_i = \phi(\mathbf{x_i})$.
- As $y = y(\phi(\mathbf{x})) \in [0,1]$ we interpret is as the probability of assigning input $\mathbf{x}$ to class 1, so that the likelihood is

$$p(\mathbf{t}|\mathbf{w}) = \prod_{i=1}^{N} y_i^{t_i}(1-y_i)^{1-t_i}$$

where $y_i = \sigma(\mathbf{w}^T \phi_i)$.

- We need to minimise minus the log-likelihood, i.e.

$$E(\mathbf{w}) = -\log p(\mathbf{t}|\mathbf{w}) = -\sum_{i=1}^{N} t_i \log y_i + (1 - t_i) \log(1 - y_i)$$

2.3 Numerical optimisation

- The gradient of $E(\mathbf{w})$ is $\nabla E(\mathbf{w}) = \sum_{i=1}^{N} (y_i - t_i)\phi_i$. The equation $\nabla E(\mathbf{w}) = 0$ has no closed form solution, so we need to solve it numerically.
- One possibility is gradient descend. We initialise $\mathbf{w}^0$ to any value and then update it by

$$\mathbf{w}^{n+1} = \mathbf{w}^n - \eta \nabla E(\mathbf{w}^n)$$

  where the method converges for η small.
- We can also use stochastic gradient descent for online training, using the update rule for $\mathbf{w}$:

$$\mathbf{w}^{n+1} = \mathbf{w}^n - \eta \nabla_{n+1} E(\mathbf{w}^n),$$

  with $\nabla_n E(\mathbf{w}) = (y_n - t_n)\phi_n$

2.4 Newton-Rapson method

- As an alternative optimisation, we can use the Newton-Rapson method, which has better convergence properties.
- The update rule reads:

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \mathbf{H}^{-1} \nabla E(\mathbf{w}^{old})$$

  where $\mathbf{H}$ is the Hessian of $E(\mathbf{w})$, and η the learning rate.
- For logistic regression, we have $\nabla E(\mathbf{w}) = \Phi^T(\mathbf{y} - \mathbf{t})$ and $\mathbf{H} = \Phi^T \mathbf{R} \Phi$, with R diagonal matrix with elements $R_{nn} = y_n(1 - y_n)$.
- It is easy to check that the Hessian is positive definite, hence the function $E(\mathbf{w})$ is convex and has a unique minimum.

2.5 Overfitting

- If we allocate each point $\mathbf{x}$ to the class with highest probability, i.e. maximising $\sigma(\mathbf{w}^T \phi(\mathbf{x}))$, then the separating surface is an hyperplane in the feature space and is given by the equation $\mathbf{w}^T \phi(\mathbf{x}) = 0$.
- If the data is linearly separable in the feature space, then any separable hyperplane is a solution, and the magnitude of $\mathbf{w}$ tends to go to infinity during optimisation. In this case, the logistic function converges to the Heaviside function.
- To avoid this issue, we can add a regularisation term to $E(\mathbf{w})$, thus minimising $E(\mathbf{w}) + \alpha \mathbf{w}^T \mathbf{w}$.

2.6 Multi-class logistic regression

- We can model directly the multiclass conditional probability, using the *soft-max function*:
$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$
with $a_k = \mathbf{w_k}\phi(\mathbf{x})$. It holds $\frac{\partial y_k(\mathbf{x})}{\partial a_j} = y_k(\delta_{kj} - y_j)$

- Using the boolean encoding of the outputs, the likelihood is
$$p(\mathbf{T}|\mathbf{w_1}, \dots, \mathbf{w_K}) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\phi_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

- Hence we need to minimise
$$E(\mathbf{w_1}, \dots, \mathbf{w_K}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log y_{nk}$$

- $E(\mathbf{w_1}, \dots, \mathbf{w_K})$ has gradient
$$\nabla_{\mathbf{w_j}} E(\mathbf{w_1}, \dots, \mathbf{w_K}) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\phi_n$$

- and Hessian with blocks given by
$$\nabla_{\mathbf{w_k}} \nabla_{\mathbf{w_j}} E(\mathbf{w_1}, \dots, \mathbf{w_K}) = -\sum_{n=1}^{N} y_{nk}(I_{kj} - y_{nj})\phi_n\phi_n^T$$

- Also in this case the Hessian is positive definite, and we can use the Newton-Rapson algorithm for optimisation

3 Laplace Approximation

3.1 One dimensional case

- It is a general technique to locally approximate a general distribution around a mode with a Gaussian.

- Consider a 1d distribution $p(z) = \frac{1}{Z}f(z)$ where $Z = \int f(z)dz$ is the normalisation constant.

- Pick a mode z_0 of $f(z)$, i.e. a point such that $\frac{d}{dz}f(z_0) = 0$.

- As the logarithm of the Gaussian density is quadratic, we consider a Taylor expansion of $\log f(z)$ around z_0:

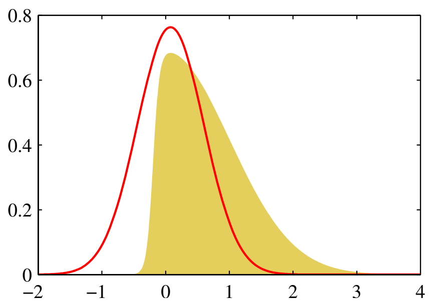
$$\log f(z) \approx \log f(z_0) - \frac{1}{2}A(z - z_0)^2$$

  with $A = -\frac{d^2}{dz^2} \log f(z_0)$

- Hence we have $f(z) \approx f(z_0)\exp(-\frac{1}{2}A(z - z_0)^2)$. Now, we seek the best Gaussian $q(z)$ approximating $p(z)$ around the model z_0, requiring $A > 0$. This is clearly given by

$$q(z) = \left(\frac{A}{2\pi}\right)^{\frac{1}{2}} \exp(-\frac{1}{2}A(z - z_0)^2)$$

- We also have that $Z \approx f(z_0)\left(\frac{A}{2\pi}\right)^{-\frac{1}{2}}$



3.2 n dimensional case

- In n dimensions, we proceed in the same way. Given a density $p(\mathbf{z}) = \frac{1}{Z}f(\mathbf{z})$, we find a mode $\mathbf{z_0}$ (so that $\nabla \log f(\mathbf{z_0}) = \mathbf{0}$, and approximate $\log f(\mathbf{z})$ around $\mathbf{z_0}$ by Taylor expansion, obtaining

$$\log f(\mathbf{z}) = \log f(\mathbf{z_0}) - \frac{1}{2}(\mathbf{z} - \mathbf{z_0})^T \mathbf{A}(\mathbf{z} - \mathbf{z_0})$$

  where $\mathbf{A} = -\nabla\nabla \log f(\mathbf{z_0})$.

- This gives a Gaussian approximation around $\mathbf{z_0}$ by

$$q(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{z_0}, \mathbf{A}^{-1})$$

- Furthermore $Z \approx \frac{(2\pi)^{n/2}}{|\mathbf{A}|^{1/2}} f(\mathbf{z_0})$

3.3 Model comparison and BIC

- We can use Laplace approximation for the marginal likelihood in a model comparison framework.
- Consider data $\mathcal{D}$ and a model $\mathcal{M}$ depending on parameters θ. We fix a prior $\mathcal{P}(\theta)$ over θ and compute the posterior by Bayes theorem:

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

- Here $p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta$ is the marginal likelihood. It fits in the previous framework by setting $Z = p(\mathcal{D})$, and $f = p(\mathcal{D}|\theta)p(\theta)$.
- By Laplace approximation around the maximum a-posteriori estimate θ_{MAP}:

$$\log p(\mathcal{D}) \approx \log p(\mathcal{D}|\theta_{MAP}) + \log p(\theta_{MAP}) + \frac{M}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{A}|$$

  where $\mathbf{A} = -\nabla\nabla p(\mathcal{D}|\theta_{MAP})p(\theta_{MAP})$. The last three terms in the sum penalise the log likelihood in terms of model complexity.
- A crude approximation of them is

$$logp(\mathcal{D}) \approx \log p(\mathcal{D}|\theta_{MAP}) - \frac{1}{2}M\log N$$

  which is known as *Bayesian Information Content*, and can be used to penalise log likelihood w.r.t. model complexity, to compare different models.

4 Bayesian Logistic Regression

4.1 The Bayesian way

- To recast logistic regression in a Bayesian framework, we need to put a prior on $p(\mathbf{w})$ of the coefficients $\mathbf{w}$ of $\sigma(\mathbf{w}^T\phi(\mathbf{x}))$ and compute the posterior distribution on $\mathbf{w}$ by Bayes theorem. Then we can make predictions by integrating out the parameters.
- Assume a Gaussian prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m_0}, \mathbf{S_0})$. The posterior is $p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w})p(\mathbf{t}|\mathbf{w})$, and the log-posterior is

$$\log p(\mathbf{w}|\mathbf{t}) = -\frac{1}{2}(\mathbf{w}-\mathbf{m_0})^T S_0^{-1}(\mathbf{w}-\mathbf{m_0}) + \sum_{i=1}^{N}[t_i\log y_i + (1-t_i)\log(1-y_i)] + c$$

  where $y_i = \sigma(\mathbf{w}\phi(\mathbf{x_i}))$.
- Computing the marginal likelihood and the normalisation constant is analytically intractable, due to quadratic and logistic terms. Hence we do a Laplace approximation of the posterior.

4.2 Laplace approximation of the posterior

- Given $\log p(\mathbf{w}|\mathbf{t})$, we first find the maximum a-posteriori $\mathbf{w_{MAP}}$, by running a numerical optimisation, and then obtain the Laplace approximation computing the Hessian matrix at $\mathbf{w_{MAP}}$ and inverting it, obtaining

$$\mathbf{S_N} = -\nabla\nabla \log p(\mathbf{w}|\mathbf{t}) = \mathbf{S_0}^{-1} + \sum_{n=1}^{N} y_n(1-y_n)\phi(\mathbf{x_n})\phi(\mathbf{x_n})^T$$

  evaluated at $\mathbf{w} = \mathbf{w_{MAP}}$.

- Hence, the Laplace approximation of the posterior is

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w_{MAP}}, \mathbf{S_N})$$

4.3 Predictive distribution

- The predictive distribution for class C_1 is given by

$$p(C_1|\phi, \mathbf{t}) = \int p(C_1|\phi, \mathbf{w}, \mathbf{t})q(\mathbf{w})d\mathbf{w} = \int \sigma(\mathbf{w}^T\phi(\mathbf{x}))q(\mathbf{w})d\mathbf{w}$$

- This multi-dimensional integral can be simplified by noting that it depends on $\mathbf{w}$ only on the 1-dim projection $a = \mathbf{w}^T\phi(\mathbf{x})$, and that q restricted to this direction is still a Gaussian distribution $q(a)$ with mean and variance

$$\mu_a = \mathbf{w_{MAP}}^T\phi(\mathbf{x}) \quad \sigma_a^2 = \phi(\mathbf{x})^T\mathbf{S_N}\phi(\mathbf{x})$$

- Hence we have

$$p(C_1|\phi, \mathbf{t}) = \int \sigma(a)q(a)da$$

4.4 Probit approximation

- The integral $p(C_1|\phi, \mathbf{t}) = \int \sigma(a)q(a)da$ can be approximated by approximating the logistic function by the probit: $\sigma(a) = \Psi(\lambda a)$, where λ is obtained by matching derivatives at zero and is $\lambda^2 = \pi/8$.

- We then use

$$\int \Psi(a)\mathcal{N}(a|\mu, \sigma^2) = \Psi\left(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{1/2}}\right)$$

  and approximate back to the logistic to get

$$p(C_1|\phi, \mathbf{t}) \approx \sigma(\kappa(\sigma_a^2)\mu_a)$$

  with $\kappa(\sigma_a^2) = (1 + \pi\sigma_a^2/8)^{-1/2}$

5 Constrained Optimisation

5.1 Lagrange Multipliers

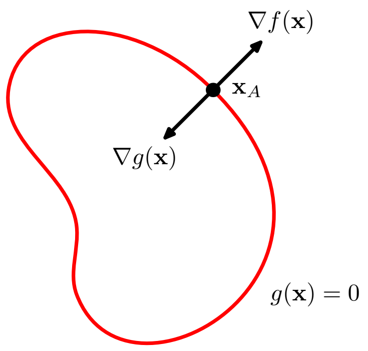
- Suppose we want to maximise $f(\mathbf{x})$ subject to the constraint $g(\mathbf{x}) = 0$.
- $g(\mathbf{x}) = 0$ defines a surface and $\nabla g(\mathbf{x})$ is always orthogonal to it.

- In a point of this surface in which $f(\mathbf{x})$ is optimal, it must hold that $\nabla f(\mathbf{x}) = \lambda \nabla g(\mathbf{x})$, i.e. the projection of $\nabla f(\mathbf{x})$ on the tangent space of the surface is zero, otherwise we could increment the value of f by moving along the surface $g(\mathbf{x}) = 0$.
- We can then do an unconstrained optimisation

$$\max_{\mathbf{x}} \inf_{\lambda} L(\mathbf{x}, \lambda)$$

of the Lagrangian function

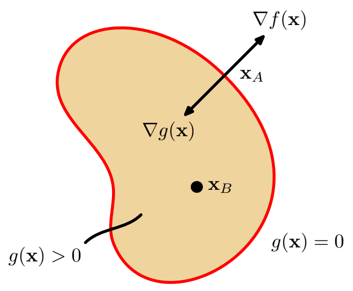
$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$$



- In fact, if $g(\mathbf{x}) \neq 0$, then $\inf_\lambda L(\mathbf{x}, \lambda) = -\infty$, hence the Lagrangian optimization problem takes finite values only on $\{g(\mathbf{x}) = 0\}$.
- Deriving w.r.t $\mathbf{x}$ gives the condition on gradients, deriving w.r.t λ the constraint: setting the derivative to zero, we enforce the constraint and look for an optimal point.

5.2 Karush-Kuhn-Tucker conditions

- Suppose we want to optimise $f(\mathbf{x})$ subject to the constraint $g(\mathbf{x}) \geq 0$.
- If an optimum $\mathbf{x}$ satisfies $g(\mathbf{x}) > 0$ (inactive constraint), then $\nabla f(\mathbf{x}) = 0$ and $\lambda = 0$, if instead $g(\mathbf{x}) = 0$ (active constraint), then $\nabla f(\mathbf{x}) = -\lambda \nabla g(\mathbf{x})$, $\lambda > 0$ because an increase of f cannot bring inside the feasible region.



- In any case $\lambda g(\mathbf{x}) = 0$ for an optimum point.

- We can then optimise the Lagrangian function $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$ subject to $\lambda \geq 0$, $g(\mathbf{x}) \geq 0$, $\lambda g(\mathbf{x}) = 0$, known as the Karush-Kuhn-Tucker (KKT) conditions.
- Also in this case, we can then solve the unconstrained optimisation

$$\max_{\mathbf{x}} \inf_{\lambda \geq 0} L(\mathbf{x}, \lambda)$$

of the Lagrangian function

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$$

- In fact, if $g(\mathbf{x}) > 0$, then the inner optimization is solved by $\lambda = 0$, otherwise, if $g(\mathbf{x}) < 0$, it is solved by $\lambda = +\infty$ and the Lagrangian is $-\infty$. On the boundary $g(\mathbf{x}) = 0$, λ can take finite values.
- To minimise $f(\mathbf{x})$, we optimise $\min_{\mathbf{x}} \sup_{\lambda \geq 0} f(\mathbf{x}) - \lambda g(\mathbf{x})$
- Lagrange and KKT multipliers can be combined to solved constrained problems with both equalities and inequalities.

5.3 The dual formulation

- The dual formulation of the constrained minimisation problem with Lagrangian $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \sum_j \lambda_j g_j(\mathbf{x})$ is given by

$$\tilde{L}(\lambda) = \inf_{\mathbf{x} \in \mathcal{D}} L(\mathbf{x}, \lambda)$$

- $\tilde{L}(\lambda)$ is a lower bound on $f(\mathbf{x})$. The dual optimisation problem is to maximise $\tilde{L}(\lambda)$ subject to KKT conditions.
- If the original problem is convex (single global optimum), and under regularity conditions on the constraints (e.g. linear), then the solution of the dual gives exactly the minimum of the primal.
- For non-convex problems, there can be a duality gap.
- For quadratic objective functions and linear constraints, the dual objective can be computed easily, because $\partial L(\mathbf{x}, \lambda)/\partial \mathbf{x}$ gives a linear system that can be solved to express $\mathbf{x}$ as a function of λ's

6 Support Vector Machines

6.1 Kernel trick for classification

- The trick works similarly as for regression. Consider class conditionals $p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$.
- We can make the assumption that $\mathbf{w} = \sum_{n=1}^{N} a_n \phi(\mathbf{x_n})$ (this is consistent, as the ML solution will belong to the space spanned by $\phi(\mathbf{x_n})$), thus getting

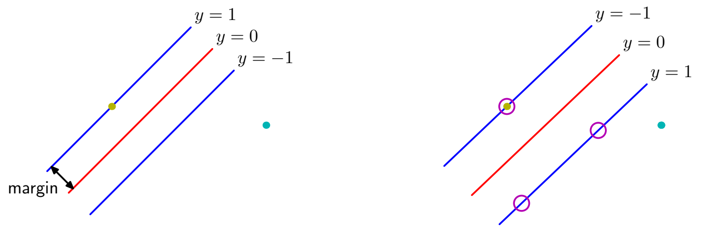
$$p(C_1|\mathbf{x}) = \sigma\left(\sum_{n=1}^{N} \alpha_n k(\mathbf{x}, \mathbf{x_n})\right)$$

where we define the kernel function $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$

- We can write also $p(C_1|\mathbf{x}) = \sigma(\mathbf{a}^T \mathbf{k}(\mathbf{x}))$. The maximum likelihood solution can be found using gradient based methods.

6.2 Maximum margin classifiers

- We have 2-class data $\mathbf{x_n}, t_n$, with $t_n \in \{-1, 1\}$. We assume for the moment that the data is linearly separable in a feature space after applying the non-linear mapping $\phi(\mathbf{x})$.
- There may be many hyperplanes separating the data. An effective choice is to select the one maximising the *margin*, i.e. the smallest distance between the separating hyperplane and the data points.
- Only closest data points are needed to determine it.



- Write $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$.
- The distance between a point and the separating hyperplane $\mathbf{w}^T\phi + b$ is $|y(\mathbf{x})|/||\mathbf{w}||$.
- As we want to classify correctly all points, it will hold that $t_n y(\mathbf{x_n}) \geq 0$, by the choice of t_n encoding.
- Hence, to find the maximum margin, we need to find $\mathbf{w}$ and b such that:

$$\max_{\mathbf{w},b}\left[\frac{1}{||\mathbf{w}||}\min_n\{t_n\mathbf{w}^T\phi(\mathbf{x_n}) + t_n b\}\right]$$

- The solution is defined up to an arbitrary rescaling of $\mathbf{w}$ and b, so we can set to 1 the margin, obtaining the constraint

$$t_n\mathbf{w}^T\phi(\mathbf{x_n}) + t_n b \geq 1, \quad n = 1, \dots, N$$

- The constraints $t_n\mathbf{w}^T\phi(\mathbf{x_n}) + t_n b \geq 1$ known as the canonical representation. Points for which equality to 1 holds are called active, the others inactive.
- The maximisation above is equivalent to minimise $||\mathbf{w}||^2$:

$$\min_{\mathbf{w},b}\frac{1}{2}||\mathbf{w}||^2$$

subject to canonical constraints. b will be set via the constraints.

- To solve this quadratic program, we introduce a Langrange multiplier a_n for each constraint, resulting in the following Lagrangian

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} a_n[t_n\mathbf{w}^T\phi(\mathbf{x_n}) + t_n b - 1]$$

which has to be minimised w.r.t $\mathbf{w}$ and b, and maximised w.r.t $\mathbf{a}$.

6.3 The dual formulation of the maximum margin problem

- Starting from the Lagrangian $L(\mathbf{w}, b, \mathbf{a})$ we compute derivatives w.r.t. $\mathbf{w}$ and b and set them to zero, obtaining constraints

$$\mathbf{w} = \sum_n a_n t_n \phi(\mathbf{x_n}) \qquad 0 = \sum_n a_n t_n$$

- By substituting them in the Lagrangian, we obtain the dual representation

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x_n}, \mathbf{x_m})$$

  subject to the constraints

$$a_n \geq 0,\ n = 1, \ldots, N; \quad \sum_n a_n t_n = 0$$

- $k(\mathbf{x_n}, \mathbf{x_m}) = \phi(\mathbf{x_n})^T \phi(\mathbf{x_m})$ is the kernel function.

6.3.1 The dual formulation of the maximum margin problem

- This optimisation problem can be solved in $O(N^3)$ time. Its main advantage is that it depends on the kernel, not on basis functions, hence it can be applied to more general kernels.
- The prediction for a new point $\mathbf{x}$ is obtained by using the dual formulation of $\mathbf{w}$, giving

$$y(\mathbf{x}) = \sum_n a_n t_n k(\mathbf{x}, \mathbf{x_n}) + b$$

6.3.2 Sparsity of the solution

- The optimisation problem satisfies the KKT conditions:

$$a_n \geq 0; \quad t_n y(\mathbf{x_n}) - 1 \geq 0; \quad a_n[t_n y(\mathbf{x_n}) - 1] = 0$$

- This implies that either $t_n y(\mathbf{x_n}) = 1$ (the vector $\mathbf{x_n}$ is at minimum distance from the margin) or $a_n = 0$ (it does not contribute to the predictions).
- Let us indicate with $\mathcal{S}$ the set of support vectors.

6.3.3 Determining b

- From any $\mathbf{x_n} \in \mathcal{S}$, by using $t_n y(\mathbf{x_n}) = 1$, we can determine b by solving

$$t_n \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x_n}, \mathbf{x_m}) + t_n b = 1$$

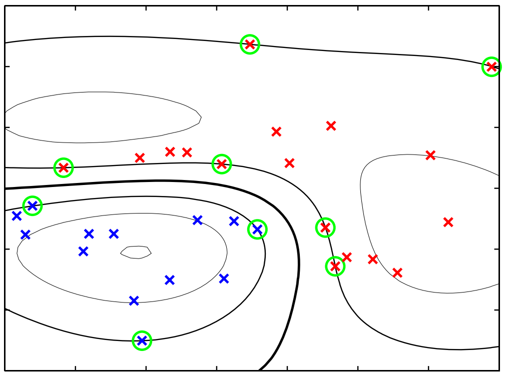
- To have a more stable solution, one multiplies by t_n, uses $t_n^2 = 1$, and averages for the different support vectors:

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x_n}, \mathbf{x_m}) \right)$$

6.4 Example of SVM

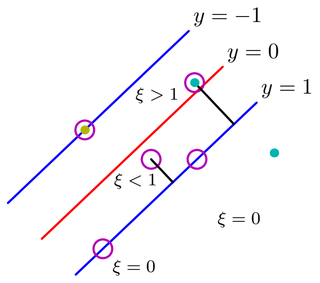
- Example of data linearly separable in the space defined by the Gaussian kernel function.
- Sparsity: only support vectors define the maximum margin hyperplane: moving the other is irrelevant, as far as they remain on the same side.

Example of synthetic data from two classes in two dimensions showing contours of constant $y(\mathbf{x})$ obtained from a support vector machine having a Gaussian kernel function. Also shown are the decision boundary, the margin boundaries, and the support vectors.



6.5 Soft margin SVM

- If class conditionals overlap, then an exact (non-linear) separation of training data may result in poor generalisation. It is better to allow some training points to be misclassified, by relaxing the constraint $t_n y(\mathbf{x_n}) \geq 1$
- We will do this by introducing N new *slack variables* $\xi_n \geq 0$, rewriting constraint as $t_n y(\mathbf{x_n}) \geq 1 - \xi_n$.
- For points correctly classified and inside the margin, we have $\xi_n = 0$, while for other points we have $\xi_n = |t_n - y(\mathbf{x_n})|$. It follows that misclassified points will have $\xi_n > 1$, while $\xi_n = 1$ only if a point lies in the separating hyperplane.
- $\sum_n \xi_n$ is an upper bound on misclassified training points.



- The primal objective function is modified to penalise the number of misclassified points:

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2}||\mathbf{w}||^2$$

- C is a regularisation term: it controls the trade-off between correct classification of training points and model complexity. For $C \to \infty$, we recover the previous SVM.
- The Lagrangian $L(\mathbf{w}, b, \mathbf{a}, \mu)$ is now given by

$$C \sum_{n-1}^{N} \xi_n + \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} a_n [t_n \mathbf{w}^T \phi(\mathbf{x_n}) + t_n b - 1 + \xi_n] - \sum_{n=1}^{N} \mu_n \xi_n$$

  with a_n, μ_n Lagrange multipliers. We omit the KKT conditions.

6.5.1 Dual formulation

- By taking partial derivatives w.r.t $\mathbf{w}$, b, and ξ_n, we obtain the dual formulation:

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x_n}, \mathbf{x_m})$$

  which has to satisfy the following box constraints

$$0 \leq a_n \leq C, \; n = 1, \ldots, N; \quad \sum_n a_n t_n = 0$$

- In the solution, we can have $a_n = 0$ (points inside the margin , for which $\xi_n = 0$), $0 < a_n < C$ (points on the margin, for which $\xi_n = 0$), or $a_n = C$ (points on the wrong side of the margin, $\xi_n > 0$).
- b can be determined as for the hard margin case, by restricting to support vectors on the margin.

6.6 SVM: comments

- The quadratic problem is convex, hence has a unique minimum, but a classic optimisation can be challenging for large problems (N large). Specialised methods have been developed, that try to decompose the problem into simpler pieces. E.g. Sequential minimal optimisation works by optimising two a_n's at time.
- SVM are hard to generalise to multi-class problems (one-versus-the-rest approach being the typical approach)
- SVM do not have a probabilistic interpretation, and some ad-hoc processing is required.
- SVM can be quite sensitive to outliers (misclassified points deeply inside the other's class region).