Linear Models

Prof. Dr. H. H. Takada

Quantitative Research – Itaú Asset Management
Institute of Mathematics and Statistics – University of São Paulo

Fitting a Straight Line

Given training data $(x^n, y^n), n = 1, ..., N$, for scalar input x^n and scalar output y^n, a linear regression fit is

$$y(x) = a + bx.$$

The sum squared training error is

$$E(a,b) = \sum_{n=1}^{N} [y^n - y(x^n)]^2 = \sum_{n=1}^{N} (y^n - a - bx^n)^2 .$$

The minimization of $E(a,b)$ is called ordinary least squares (OLS). Differentiating with respect to a and b:

$$\frac{\partial}{\partial a} E(a,b) = -2 \sum_{n=1}^{N} (y^n - a - bx^n) , \frac{\partial}{\partial b} E(a,b) = -2 \sum_{n=1}^{N} (y^n - a - bx^n) x^n$$

Dividing by N and equating to zero,

$$\langle y \rangle - a - b\langle x \rangle = 0, \langle xy \rangle - a\langle x \rangle - b\langle x^2 \rangle = 0$$

Then,

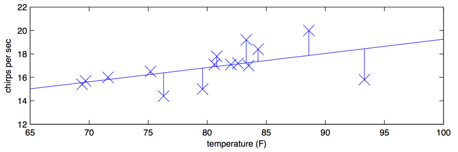
$$a = \langle y \rangle - b\langle x \rangle$$

$$b\langle x^2 \rangle = \langle xy \rangle - \langle x \rangle(\langle y \rangle - b\langle x \rangle) \Rightarrow b[\langle x^2 \rangle - \langle x \rangle^2] = \langle xy \rangle - \langle x \rangle\langle y \rangle$$

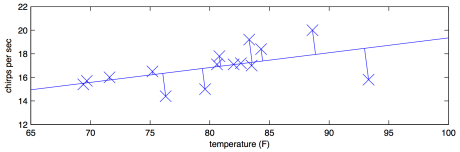
$$b = \frac{\langle xy \rangle - \langle x \rangle\langle y \rangle}{\langle x^2 \rangle - \langle x \rangle^2}.$$

In contrast to *ordinary least squares* regression, PCA minimizes the orthogonal projection of y to the line and is known as *orthogonal least squares*.

Example



(a)



(b)

Figure: Data from crickets - the number of chirps per second, versus the temperature in Fahrenheit. (a): Straight line regression fit to the cricket data. (b): PCA fit to the data. In regression we minimize the residuals - the vertical distances from datapoints to the line. In PCA the fit minimizes the orthogonal projections to the line. In this case, there is little difference in the fitted lines. Both go through the mean of the data; the linear regression fit has slope 0.121 and the PCA fit has slope 0.126.

Linear Parameter Models (LPM) for Regression

For a dataset $\{(\mathbf{x}^n, y^n), n = 1, ..., N\}$, a LPM is

$$y(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}),$$

where $\phi(\mathbf{x})$ is a vector valued function of the input vector $\mathbf{x}$.
The sum squared training error is

$$E(\mathbf{w}) = \sum_{n=1}^{N}(y^n - \mathbf{w}^{\mathsf{T}}\phi^n)^2,$$

where $\phi^n \equiv \phi(\mathbf{x}^n)$.
The OLS solution is

$$\mathbf{w} = \left(\sum_{n=1}^{N}\phi^n(\phi^n)^{\mathsf{T}}\right)^{-1}\sum_{n=1}^{N}y^n\phi^n.$$

Although the solution is using matrix inversion, in practice, one finds the numerical solution using Gauss elimination - it is faster and more numerically stable.

Example

A cubic polynomial model is given by

$$y(x) = w_1 + w_2 x + w_3 x^2 + w_4 x^3.$$

The LPM is expressed using

$$\phi(x) = (1, x, x^2, x^3)^\top.$$

Matlab

```
>> setup;
>> demoCubicPoly;
```

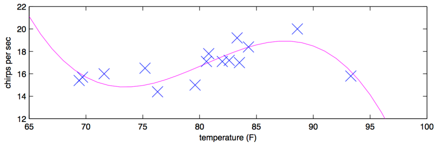


Figure: Cubic polynomial fit to the cricket data.

Vector Outputs

Consider vector outputs $\mathbf{y}$. Using a separate weight vector $\mathbf{w}_i$ for each output component y_i,

$$y_i(\mathbf{x}) = \mathbf{w}_i^{\mathsf{T}} \phi(\mathbf{x}).$$

The mathematics follows similarly to before, and we may define a train error per output as

$$E(\mathbf{w}) = \sum_i E(\mathbf{w}_i) = \sum_i \sum_n (y_i^n - \mathbf{w}_i^{\mathsf{T}} \phi^n)^2.$$

Since the training error decomposes into individual terms, one for each output, the weights for each output can be trained separately. In other words, the problem decomposes into a set of independent scalar output problems. In case the parameters $\mathbf{w}$ are tied or shared amongst the outputs, the training is still straightforward since the objective function remains linear in the parameters.

Regularization

For most purposes, our interest is not just to find the function that best fits the training data but one that that will generalize well. To control the complexity of the fitted function we may add an extra regularizing term to the training error to penalize rapid changes in the output.

LPM

The regularized sum squared training error is

$$E(\mathbf{w}) = \sum_{n=1}^{N} (y^n - \mathbf{w}^{\mathsf{T}} \phi^n)^2 + \mathbf{w}^{\mathsf{T}} \mathbf{R} \mathbf{w},$$

where

$$\mathbf{R} = \lambda \sum_{n=1}^{N} \sum_{n'=1}^{N} e^{-\gamma(\mathbf{x}^n - \mathbf{x}^{n'})^2} (\phi^n - \phi^{n'})(\phi^n - \phi^{n'})^{\mathsf{T}}.$$

The optimal $\mathbf{w}$ is given by

$$\mathbf{w} = \left(\sum_n \phi^n (\phi^n)^{\mathsf{T}} + \mathbf{R} \right)^{-1} \sum_n y^n \phi^n.$$

In practice, it is common to use a regularizer that penalizes the sum squared length of the weights

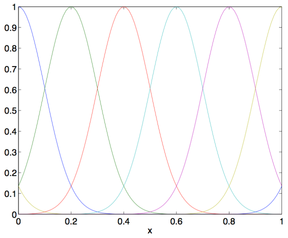
$$\lambda \mathbf{w}^{\mathsf{T}} \mathbf{w} = \lambda \sum_i w_i^2$$

which corresponds to setting $\mathbf{R} = \lambda \mathbf{I}$.

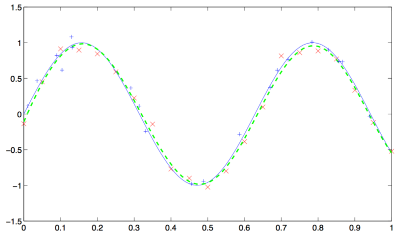
Radial Basis Functions (RBF)

A popular LPM is given by the non-linear function $\phi(\mathbf{x})$ with components

$$\phi_i(\mathbf{x}) = e^{-\frac{1}{2\alpha^2}(\mathbf{x}-\mathbf{m}^i)^2}.$$



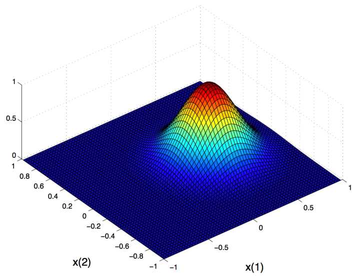
(a)



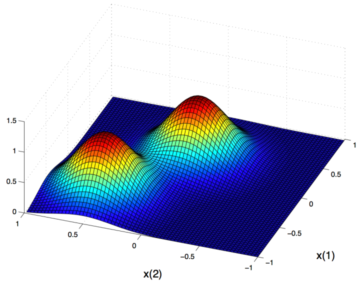
(b)

Figure: (a): A set of fixed-width ($\alpha = 1$) radial basis functions, $e^{-\frac{1}{2}(\mathbf{x}-\mathbf{m}^i)^2}$, with the centres mi evenly spaced. By taking a linear combination of these functions we can form a flexible function class. (b): The $\times$ are the training points, and the $+$ are the validation points. The solid line is the correct underlying function $\sin(10x)$ which is corrupted with a small amount of additive noise to form the train data. The dashed line is the best predictor based on the validation set.

Radial Basis Functions (RBF)



(a)



(b)

Figure: (a) The output of an RBF function $e^{-\frac{1}{2\alpha^2}(\mathbf{x}-\mathbf{m}^1)^2}$. Here, $\mathbf{m}^1 = (0, 0.3)^{\mathsf{T}}$ and $\alpha = 0.25$. (b) The combined output for two RBFs with $\mathbf{m}^1$ as above and $\mathbf{m}^2 = (0.5, -0.5)^{\mathsf{T}}$.

The Dual Representation and Kernels

Consider a set of training data with inputs, $\mathcal{X} = \{\mathbf{x}^n, n = 1, ..., N\}$ and corresponding outputs $y^n, n = 1, ..., N$. For a LPM of the form

$$f(\mathbf{x}) = \mathbf{w}^{\top}\mathbf{x},$$

we assume that we have found an optimal parameter $\mathbf{w}_*$.

The nullspace of $\mathcal{X}$ are those $\mathbf{x}^{\perp}$ which are orthogonal to all the inputs in $\mathcal{X}$:

$$(\mathbf{x}^{\perp})^{\top}\mathbf{x}^n = 0, \forall n = 1, ..., N.$$

Then,

$$(\mathbf{w}_* + \mathbf{x}^{\perp})^{\top}\mathbf{x}^n = \mathbf{w}_*^{\top}\mathbf{x}^n.$$

This means that adding a contribution to $\mathbf{w}_*$ outside of the space spanned by $\mathcal{X}$, has no effect on the predictions on the train data. If the training criterion depends only on how well the LPM predicts the train data, there is therefore no need to consider contributions to $\mathbf{w}$ from outside of $\mathcal{X}$. (Is that ok?)

The Dual Representation and Kernels

Then, without loss of generality we may consider the representation

$$\mathbf{w} = \sum_{n=1}^{N} a_n \mathbf{x}^n.$$

The parameters $\mathbf{a} = (a_1, ..., a_N)$ are called the dual parameters. Consequently, the dual representation is

$$f(\mathbf{x}^n) = \mathbf{w}^{\mathsf{T}} \mathbf{x}^n = \sum_{m=1}^{N} a_m (\mathbf{x}^m)^{\mathsf{T}} \mathbf{x}^n.$$

The Dual Representation and Kernels

More generally,

$$\mathbf{w} = \sum_{n=1}^{N} a_n \phi(\mathbf{x}^n)$$

and we may write

$$f(\mathbf{x}^n) = \mathbf{w}^\top \phi(\mathbf{x}^n) = \sum_{m=1}^{N} a_m \phi(\mathbf{x}^m)^\top \phi(\mathbf{x}^n) = \sum_{m=1}^{N} a_m K(\mathbf{x}^m, \mathbf{x}^n),$$

where we have defined a kernel function

$$K(\mathbf{x}^m, \mathbf{x}^n) \equiv \phi(\mathbf{x}^m)^\top \phi(\mathbf{x}^n) \equiv [\mathbf{K}]_{m,n}.$$

Finally,

$$f(\mathbf{x}^n) = \mathbf{w}^\top \phi(\mathbf{x}^n) = [\mathbf{K}\mathbf{a}]_n = \mathbf{a}^\top \mathbf{k}^n,$$

where $\mathbf{k}^n$ is the nth column of the Gram matrix $\mathbf{K}$. By construction, the Gram matrix must be semidefinite, so the kernel must correspond to a covariance function.

Regression in the Dual-Space

For OLS, the train error is given by

$$E(\mathbf{a}) = \sum_{n=1}^{N} (y^n - \mathbf{a}^\top \mathbf{k}^n)^2$$

and regularization term that penalizes the sum squared length of the weights is given by

$$\mathbf{w}^\top \mathbf{w} = \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m \phi(\mathbf{x}^n) \phi(\mathbf{x}^m) = \mathbf{a}^\top \mathbf{K} \mathbf{a}.$$

Then, the optimal solution for $\mathbf{a}$ is therefore

$$\mathbf{a} = \left(\sum_{n=1}^{N} \mathbf{k}^n (\mathbf{k}^n)^\top + \lambda \mathbf{K} \right)^{-1} \sum_{n=1}^{N} y^n \mathbf{k}^n.$$

Regression in the Dual-Space

It is possible to rewrite the previous optimal solution:

$$\mathbf{a} = \left(\sum_{n=1}^{N} \mathbf{K}^{-1} \mathbf{k}^n (\mathbf{k}^n)^\top + \lambda \mathbf{I} \right)^{-1} \sum_{n=1}^{N} y^n \mathbf{K}^{-1} \mathbf{k}^n$$

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}.$$

Then, the prediction for a new input $\mathbf{x}^*$ is given by

$$y(\mathbf{x}^*) = \mathbf{k}_*^\top (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y},$$

where

$$[\mathbf{k}_*]_m = K(\mathbf{x}^*, \mathbf{x}^m).$$

This dual space solution shows that predictions can be expressed purely in terms of the kernel K. This means that we may dispense with defining the vector functions $\phi(\mathbf{x})$ and define a kernel function directly.

LPM for Classification

In a binary classification problem, we are given train data, $\mathcal{D} = \{(\mathbf{x}^n, c^n), n = 1, ..., N\}$, where the targets $c \in \{0, 1\}$. It is possible to define

$$p(c = 1|\mathbf{x}) = f(\mathbf{x}^\top \mathbf{w}),$$

where $0 \leq f(x) \leq 1$. $f(x)$ is called a mean function - the inverse function $f^{-1}(x)$ is the link function.

Logit Regression

The logistic function is a popular choice for the mean function:

$$f(x) = \sigma(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}},$$

which is also called logistic sigmoid. The scaled version is defined as

$$\sigma_\beta(x) = \sigma(\beta x) = \frac{1}{1 + e^{-\beta x}}.$$

The link function is called logit function.

LPM for Classification

Probit Regression

The cumulative standard normal distribution:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}t^2} dt = \frac{1}{2}(1 + \mathsf{erf}(x)),$$

where the error function is given by

$$\mathsf{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-t^2} dt.$$

The link function is called probit function.

Example

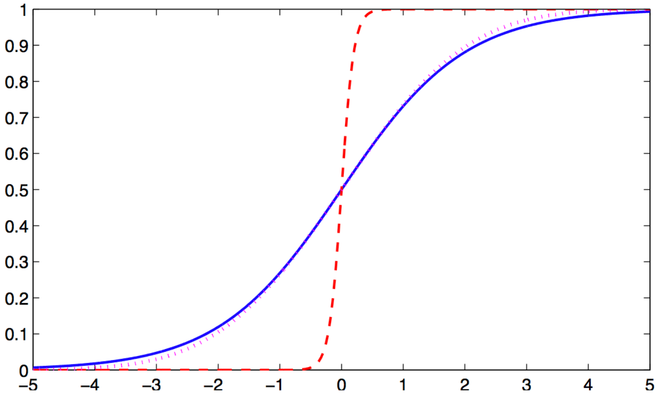


Figure: The logistic sigmoid function $\sigma_\beta(x) = 1/(1 + e^{-\beta x})$. The parameter β determines the steepness of the sigmoid. The full (blue) line is for $\beta = 1$ and the dashed (red) for $\beta = 10$. As $\beta \to \infty$, the logistic sigmoid tends to a Heaviside step function. The dotted curve (magenta) is the cumulative standard normal distribution $0.5(1 + \text{erf}(\lambda x))$ for $\lambda = \sqrt{\pi}/4$, which closely matches the logistic sigmoid with $\beta = 1$.

Logistic Regression

Generically, logistic regression corresponds to the model

$$p(c = 1|\mathbf{x}) = \sigma(b + \mathbf{x}^{\mathsf{T}}\mathbf{w}),$$

where b is a scalar.

Decision Boundary

The decision boundary is defined as that set of $\mathbf{x}$ for which $p(c = 1|\mathbf{x}) = p(c = 0|\mathbf{x}) = 0.5$. This is given by the hyperplane

$$b + \mathbf{x}^{\mathsf{T}}\mathbf{w} = 0.$$

If all the train data for class 1 lies on one side of a hyperplane, and for class 0 on the other, the data is said to be linearly separable.

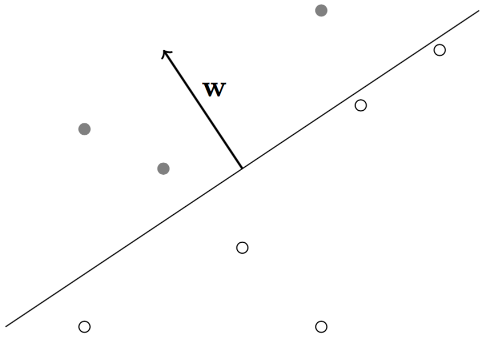


Figure: For two dimensional data, the decision boundary is a line. If all the training data for class 1 (filled circles) lie on one side of the line, and for class 0 (open circles) on the other, the data is said to be linearly separable. The orientation of the decision boundary is determined by $\mathbf{w}$, the normal to the hyperplane.

The Perceptron

The perceptron assigns $\mathbf{x}$ to class 1 if $b + \mathbf{x}^{\mathsf{T}}\mathbf{w} \geq 0$ and to class 0 otherwise:

$$p(c = 1|\mathbf{x}) = \theta(b + \mathbf{x}^{\mathsf{T}}\mathbf{w}),$$

where the step function is defined as

$$\theta(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}.$$

Consider the logistic regression model $p(c = 1|\mathbf{x}) = \sigma_\beta(b + \mathbf{x}^{\mathsf{T}}\mathbf{w})$ and take the limit $\beta \to \infty$. Then, it follows theperceptron like (or probabilistic perceptron) classifier

$$p(c = 1|\mathbf{x}) = \begin{cases} 1 & b + \mathbf{x}^{\mathsf{T}}\mathbf{w} > 0 \\ 0.5 & b + \mathbf{x}^{\mathsf{T}}\mathbf{w} = 0 \\ 0 & b + \mathbf{x}^{\mathsf{T}}\mathbf{w} < 0 \end{cases}.$$

The only difference between this probabilistic perceptron and the standard perceptron is in the definition of the value of the step function at 0.

Maximum Likelihood Training

Under the standard i.i.d. assumption, the likelihood of the observed data $\mathcal{D}$ is

$$p(\mathcal{D}|b, \mathbf{w}) = \prod_{n=1}^{N} p(c^n|\mathbf{x}^n, b, \mathbf{w}) p(\mathbf{x}^n)$$

$$= \prod_{n=1}^{N} p(c = 1|\mathbf{x}^n, b, \mathbf{w})^{c^n} (1 - p(c = 1|\mathbf{x}^n, b, \mathbf{w}))^{1-c^n} p(\mathbf{x}^n),$$

where we have used the fact that $c^n \in \{0, 1\}$. The loglikelihood for logistic regression is

$$L(\mathbf{w}, b) = \sum_{n=1}^{N} c^n \log[\sigma(b + \mathbf{w}^\mathsf{T}\mathbf{x}^n)] + (1 - c^n) \log[1 - \sigma(b + \mathbf{w}^\mathsf{T}\mathbf{x}^n)].$$

We need to maximize $L(\mathbf{w}, b)$...

Gradient Ascent Algorithm

It is possible to show that

$$\nabla_{\mathbf{w}} L = \sum_{n=1}^{N} (c^n - \sigma(b + \mathbf{w}^\mathsf{T} \mathbf{x}^n))\mathbf{x}^n$$

and

$$\frac{dL}{db} = \sum_{n=1}^{N} (c^n - \sigma(b + \mathbf{w}^\mathsf{T} \mathbf{x}^n)).$$

The gradient ascent procedure then corresponds to updating the weights and bias using

$$\mathbf{w}^{new} = \mathbf{w} + \eta_{\mathbf{w}} \nabla_{\mathbf{w}} L, \quad b^{new} = b + \eta_b \frac{dL}{db},$$

where $\eta_{\mathbf{w}}$ and η_b are small positive scalars called learning rates. The application of the above rule will lead to a gradual increase in the loglikelihood. It is possible to show that the loglikelihood is concave.

Batch Training

The updating rules, batch updates, (from previous slide) are:

$$\mathbf{w}^{new} = \mathbf{w} + \eta_{\mathbf{w}} \sum_{n=1}^{N} (c^n - \sigma(b + \mathbf{w}^{\top}\mathbf{x}^n))\mathbf{x}^n$$

and

$$b^{new} = b + \eta_b \sum_{n=1}^{N} (c^n - \sigma(b + \mathbf{w}^{\top}\mathbf{x}^n)).$$

Matlab

```
>> setup;
>> demoLogReg;
```

Multiple Classes

For more than two classes, one may use the softmax function

$$p(c = i|\mathbf{x}) = \frac{e^{b_i + \mathbf{w}_i^\top \mathbf{x}}}{\sum_{j=1}^{C} e^{b_j + \mathbf{w}_j^\top \mathbf{x}}},$$

where C is the number of classes. When $C = 2$ this can be reduced to the logistic sigmoid model. It is possible to show that the loglikelihood is concave.

Improving the Classification

A drawback of logistic regression as described above is the simplicity of the decision surface - a hyperplane. It is possible to map the inputs $\mathbf{x}$ in a non-linear way to $\phi(\mathbf{x})$:

$$p(c = 1|\mathbf{x}) = \sigma(b + \mathbf{w}^{\mathsf{T}}\phi(\mathbf{x})).$$

For the Maximum Likelihood criterion, we may use exactly the same algorithm as before on replacing $\mathbf{x}$ with $\phi(\mathbf{x})$.

Matlab

Logistic regression $p(c = 1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}))$ using a quadratic function $\phi(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_2^2, x_1x_2)^{\mathsf{T}}$ and 1000 iterations of gradient ascent training with learning rates $\eta_{\mathbf{w}} = \eta_b = 0.1$.

```
>> setup;
>> demoLogRegNonLinear;
```

Support Vector Machines (SVM)

In the SVM literature it is common to use $+1$ and -1 to denote the two classes. For a hyperplane defined by weight $\mathbf{w}$ and bias b, a linear discriminant is given by

$$\mathbf{w}^\top \mathbf{x} + b \begin{cases} \geq 0 & \text{class } +1 \\ < 0 & \text{class } -1 \end{cases} .$$

To make the classifier more robust, it is possible to use a finite positive amount ϵ^2:

$$\mathbf{w}^\top \mathbf{x} + b \begin{cases} \geq \epsilon^2 & \text{class } +1 \\ < -\epsilon^2 & \text{class } -1 \end{cases} .$$

Since $\mathbf{w}$, b and ϵ^2 can all be rescaled arbitrary, we set $\epsilon = 1$. Then, a point $\mathbf{x}_+$ from class $+1$ at decision boundary satisfies

$$\mathbf{w}^\top \mathbf{x}_+ + b = 1$$

and a point $\mathbf{x}_-$ from class -1 that is closest to the decision boundary satisfies

$$\mathbf{w}^\top \mathbf{x}_- + b = -1.$$

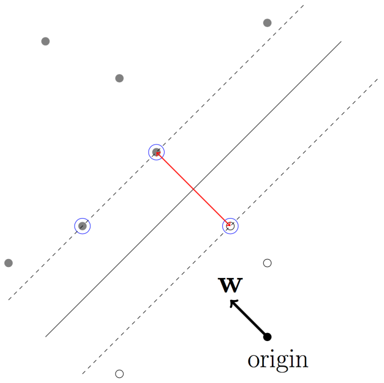


Figure: SVM classification of data from two classes (open circles and filled circles). The decision boundary $\mathbf{w}^{\mathsf{T}}\mathbf{x} + b = 0$ (solid line). For linearly separable data the maximum margin hyperplane is equidistant from the closest opposite class points. These support vectors are highlighted in blue and the margin in red. The distance of the decision boundary from the origin is $b/\sqrt{\mathbf{w}^{\mathsf{T}}\mathbf{w}}$ and the distance of a general point $\mathbf{x}$ from the origin along the direction $\mathbf{w}$ is $\mathbf{x}^{\mathsf{T}}\mathbf{w}/\sqrt{\mathbf{w}^{\mathsf{T}}\mathbf{w}}$.

The margin between the hyperplanes for the two classes is then the difference between the two distances along the direction $\mathbf{w}$ which is

$$\frac{\mathbf{w}^{\top}}{\sqrt{\mathbf{w}^{\top}\mathbf{w}}}(\mathbf{x}_{+} - \mathbf{x}_{-}) = \frac{2}{\sqrt{\mathbf{w}^{\top}\mathbf{w}}}.$$

We set the distance between the two hyperplanes to be maximal in a quadratic programming program:

$$\min \frac{1}{2}\mathbf{w}^{\top}\mathbf{w} \text{ s.t. } y^{n}(\mathbf{w}^{\top}\mathbf{x}^{n} + b) \geq 1, n = 1, ..., N.$$

For robustness purposes, it is possible to set

$$y^{n}(\mathbf{w}^{\top}\mathbf{x}^{n} + b) \geq 1 - \xi^{n},$$

where $\xi^{n} \geq 0$ are called slack variables. For $\xi^{n} > 1$, the datapoint is assigned the opposite class to its training label.

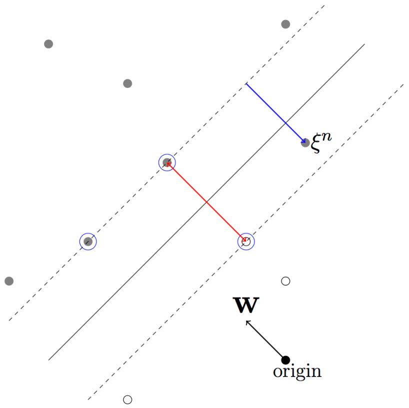


Figure: Slack margin. The term ξ^n measures how far a variable is from the wrong side of the mar- gin for its class. If $\xi^n > 1$ then the point will be misclassified and treated as an outlier.

2-norm soft-margin

The 2-norm soft-margin optimization problem is

$$\min \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} + \frac{C}{2}\sum_{n}(\xi^n)^2 \text{ s.t. } y^n(\mathbf{w}^{\mathsf{T}}\mathbf{x}^n + b) \geq 1 - \xi^n, n = 1, ..., N,$$

where C controls the number of mislabellings of the train data. The constant C needs to be determined empirically using a validation set. The Lagrangian is

$$L(\mathbf{w}, b, \xi, \alpha) = \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} + \frac{C}{2}\sum_{n}(\xi^n)^2 - \sum_{n}\alpha^n[y^n(\mathbf{w}^{\mathsf{T}}\mathbf{x}^n + b) - 1 + \xi^n],$$

where $\alpha^n \geq 0$.

Then,

$$\frac{\partial}{\partial w_i} L(\mathbf{w}, b, \xi, \alpha) = w_i - \sum_n \alpha^n y^n x_i^n = 0$$

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \xi, \alpha) = -\sum_n \alpha^n y^n = 0$$

$$\frac{\partial}{\partial \xi^n} L(\mathbf{w}, b, \xi, \alpha) = C\xi^n - \alpha^n = 0.$$

Consequently,

$$\mathbf{w} = \sum_n \alpha^n y^n \mathbf{x}^n.$$

For points $\mathbf{x}^n$ on the 'correct side' of the decision boundary

$$y^n(\mathbf{w}^\top \mathbf{x}^n + b) - 1 + \xi^n > 0,$$

maximizing L with respect to α requires the corresponding α^n to be set to zero. Only training points that are support vectors lying on the decision boundary have non-zero α^n.

Since only the support vectors have non-zero α^n, the solution for $\mathbf{w}$ will typically depend on only a small number of the training data. Using these conditions and substituting back into the original problem,

$$\max \sum_n \alpha^n - \frac{1}{2}\sum_{n,m} y^n y^m \alpha^n \alpha^m (\mathbf{x}^n)^\mathsf{T} \mathbf{x}^m - \frac{1}{2C}\sum_n (\alpha^n)^2$$

subject to

$$\sum_n y^n \alpha^n = 0, \quad \alpha^n \geq 0.$$

If $K(\mathbf{x}^n, \mathbf{x}^m) \equiv (\mathbf{x}^n)^\mathsf{T} \mathbf{x}^m$, the optimization problem becomes

$$\max \sum_n \alpha^n - \frac{1}{2}\sum_{n,m} y^n y^m \alpha^n \alpha^m \left(K(\mathbf{x}^n, \mathbf{x}^m) + \frac{1}{C}\delta_{n,m} \right)$$

subject to

$$\sum_n y^n \alpha^n = 0, \quad \alpha^n \geq 0.$$

1-norm soft-margin (box constraint)

The 1-norm soft-margin (box constraint) optimization problem is

$$\min \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} + C\sum_{n}\xi^{n} \text{ s.t. } y^{n}(\mathbf{w}^{\mathsf{T}}\mathbf{x}^{n} + b) \geq 1 - \xi^{n}, n = 1, ..., N.$$

The Lagrangian is

$$L(\mathbf{w}, b, \xi, \alpha, \mathbf{r}) = \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} + C\sum_{n}\xi^{n} - \sum_{n}\alpha^{n}[y^{n}(\mathbf{w}^{\mathsf{T}}\mathbf{x}^{n} + b) - 1 + \xi^{n}] - \sum_{n} r^{n}\xi^{n},$$

where $r^{n} \geq 0$ are introduced in order to give a non-trivial solution (otherwise $\alpha^{n} = C$).

Following similar steps as for the 2-norm case, the optimization problem is

$$\max \sum_n \alpha^n - \frac{1}{2} \sum_{n,m} y^n y^m \alpha^n \alpha^m K(\mathbf{x}^n, \mathbf{x}^m)$$

subject to

$$\sum_n y^n \alpha^n = 0, \quad 0 \leq \alpha^n \leq C.$$

SVM and Kernels

It is possible to use the following definition for the kernel function to make a non-linear classifier:

$$K(\mathbf{x}^n, \mathbf{x}^m) \equiv \phi(\mathbf{x}^n)^{\mathsf{T}} \phi(\mathbf{x}^m).$$

Matlab: A non-linear SVM

```
>> setup; demoSVM;
```

The solid red and solid blue circles represent train data from different classes. The support vectors are highlighted in green. For the unfilled test points, the class assigned to them by the SVM is given by the colour.