Support Vector Machines

Note: Unless otherwise noted all references including images are from the required textbook, Machine Learning: A Probabilistic Perspective by Kevin P. Murphy.

Support Vector Machines

One approach to derive a sparse kernel machine is to change the objective function from negative log likelihood to some other loss function. In particular consider the following function

$$J(\mathbf{w}, \lambda) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \lambda ||\mathbf{w}||^2$$

where $\hat{y}_i = \mathbf{w}^T \mathbf{x}_i + w_0$. If L is quadratic loss, this is equivalent to ridge regression.

Support Vector Machines

In the ridge regression case, the solution to this has the form $\mathbf{w} = \mathbf{X}^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{y}$, and we can rewrite these equations in a way that only involves inner products of the form $\langle \mathbf{x}, \mathbf{x}' \rangle$, which we can replace by calls to a kernel function, $\kappa(\mathbf{x}, \mathbf{x}')$. This is kernelized, but not sparse.

Support Vector Machines

If we replace the quadratic/log-loss with some other loss function, we can ensure that the solution is sparse, so that predictions only depend on a subset of the training data, known as **support vectors**. This combination of the kernel trick plus a modified loss function is known as a **support vector machine** or **SVM**.

SVMs for Regression

Consider the **epsilon insensitive loss function**, defined by

$$L_\epsilon(y, \hat{y}) \triangleq \begin{cases} 0 & \text{if } |y - \hat{y}| < \epsilon \\ |y - \hat{y}| - \epsilon & \text{otherwise} \end{cases}$$

This means that any point lying inside an ϵ-**tube** around the prediction is not penalized.

SVMs for Regression

The corresponding objective function is usually written in the following form

$$J = C\sum_{i=1}^{N} L_{\epsilon}(y_i, \hat{y}_i) + \frac{1}{2}||\mathbf{w}||^2$$

where $\hat{y}_i = f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + w_0$ and $C = 1/\lambda$ is a regularization constant. But, this objective function is not differentiable, because of the absolute value function in the loss term.

SVMs for Regression

There are several possible algorithms to solve for **w**. One approach is to formulate the problem as a constrained optimization problem using slack variables, which results in the following optimal solution

$$\hat{\mathbf{w}} = \sum_i \alpha_i \mathbf{x}_i$$

where $\alpha_i \geq 0$.

SVMs for Regression

It turns out that the $\boldsymbol{\alpha}$ vector is sparse, because we don't care about errors which are smaller than ϵ. The $\mathbf{x}_i$ for which $\alpha_i > 0$ are called the **support vectors**; these are points for which the errors lie on or outside the ϵ-**tube**.

Once the model is trained, we can then make predictions using

$$\hat{y}(\mathbf{x}) = \hat{w}_0 + \hat{\mathbf{w}}^T \mathbf{x}$$

SVMs for Regression

Plugging in the definition of $\hat{\mathbf{w}}$ we get

$$\hat{y}(\mathbf{x}) = \hat{w}_0 + \sum_i \alpha_i \mathbf{x}_i^T \mathbf{x}$$

Finally, we can replace $\mathbf{x}_i^T \mathbf{x}$ with $\kappa(\mathbf{x}_i, \mathbf{x},)$ to get a kernelized solution:

$$\hat{y}(\mathbf{x}) = \hat{w}_0 + \sum_i \alpha_i \kappa(\mathbf{x}_i, \mathbf{x})$$

SVMs for Classification

Consider binary logistic regression, and let $y_i \in \{-1, +1\}$. Suppose our decision function $f(\mathbf{x}_i)$ computes the log-odds ratio

$$f(\mathbf{x}_i) = \log \frac{p(y=1|\mathbf{x}_i, \mathbf{w})}{p(y=-1|\mathbf{x}_i, \mathbf{w})} = \mathbf{w}^T \mathbf{x}_i = \eta_i$$

Then the corresponding probability distribution on the output label is

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \text{sigm}(y_i \eta_i)$$

SVMs for Classification

Let us define the **log-loss** as

$$L_{\text{nll}}(y, \eta) = -\log p(y|\mathbf{x}, \mathbf{w}) = \log(1 + e^{-y\eta})$$

Minimizing the average log-loss is equivalent to maximizing the likelihood.

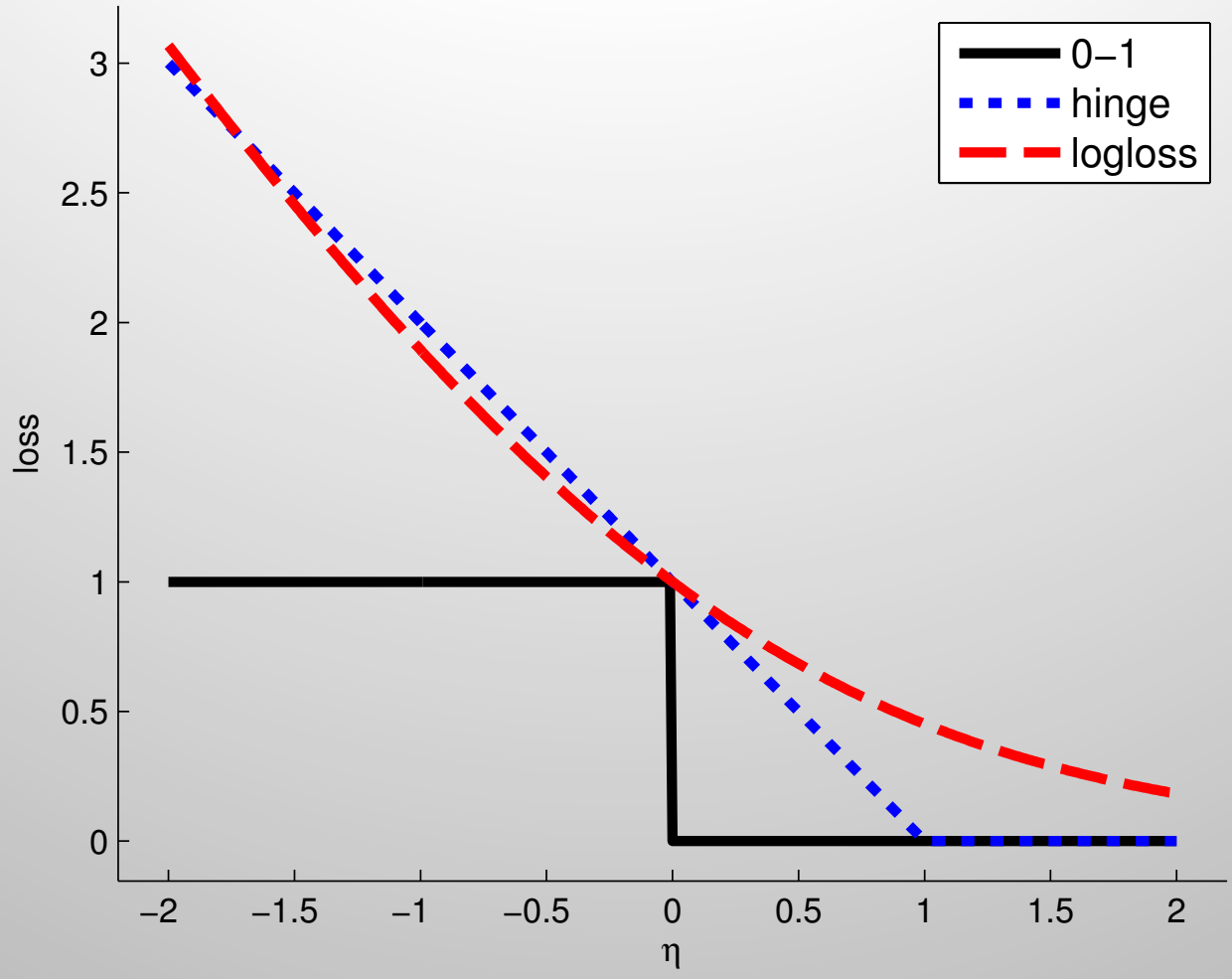
SVMs for Classification

Now we replace the NLL loss with the hinge loss, defined as

$$L_{\text{hinge}}(y, \eta) = \max(0, 1 - y\eta) = (1 - y\eta)_+$$

This function is also non-differentiable because of the max term.

SVMs for Classification



SVMs for Classification

However, by introducing slack variables, solution has the form

$$\hat{\mathbf{w}} = \sum_i \alpha_i \mathbf{x}_i$$

where $\alpha_i = \lambda_i y_i$ and where $\boldsymbol{\alpha}$ vector is sparse because of the hinge loss. The $\mathbf{x}_i$ for which $\alpha_i > 0$ are called the support vectors.

SVMs for Classification

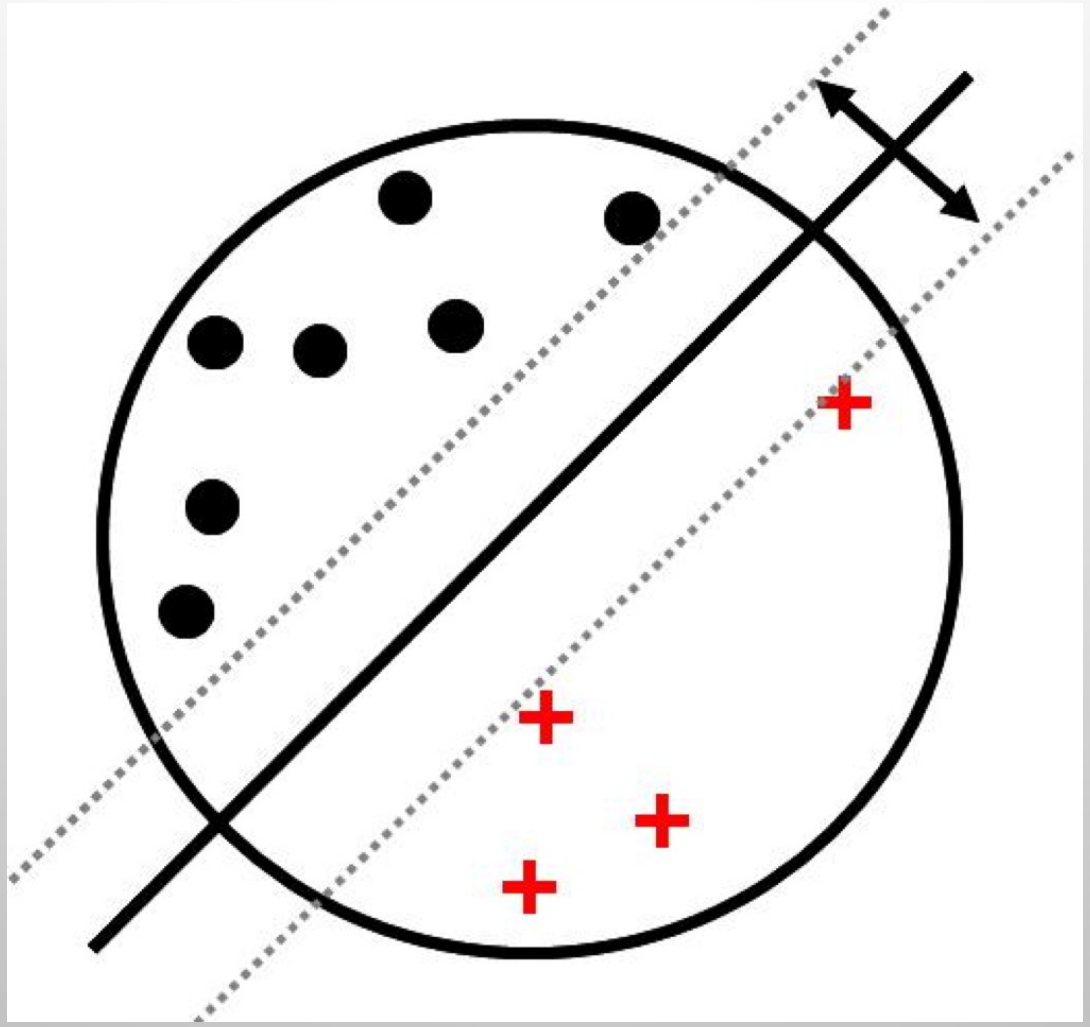
At test time, prediction is done using

$$\hat{y}(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) = \text{sgn}\left(\hat{w}_0 + \hat{\mathbf{w}}^T\mathbf{x}\right)$$

Using the kernel trick we have

$$\hat{y}(\mathbf{x}) = \text{sgn}\left(\hat{w}_0 + \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x})\right)$$

The Large Margin Principle



A separating hyper-plane with large margin

The Large Margin Principle

There might be many lines that perfectly separate the training data (especially if we work in a high dimensional feature space), but intuitively, the best one to pick is the one that maximizes the margin, i.e., the perpendicular distance to the closest point.

The Large Margin Principle

In addition, we want to ensure each point is on the correct side of the boundary, hence we want $f(\mathbf{x}_i)y_i > 0$ for all i.

So our objective becomes

$$\max_{\mathbf{w},w_0} \min_{i=1}^{N} \frac{y_i(\mathbf{w}^T\mathbf{x}_i + w_0)}{||\mathbf{w}||}$$

The Large Margin Principle

Note that by rescaling the parameters using $\mathbf{w} \rightarrow k\mathbf{w}$ and $w_0 \rightarrow kw_0$, we do not change the distance of any point to the boundary, since the k factor cancels out when we divide by $||w||$.

Therefore let us define the scale factor such that $y_i f_i = 1$ for the point that is closest to the decision boundary. We therefore want to optimize

$$\min_{\mathbf{w}, w_0} \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1, i = 1 : N$$

Choosing C

SVMs for both classification and regression require that you specify the kernel function and the parameter C. Typically C is chosen by cross-validation. Note, however, that C interacts quite strongly with the kernel parameters.