

1-class classification (anomaly detection)

There are a lot of important problems such as monitoring where we essentially only have positive examples. For instance, it would be hard to generate “negative” examples (failures) when monitoring the operation of a large manufacturing system. The problem is also called *anomaly detection* as negative examples represent deviations from the typical behavior.

Another problem of this type is intrusion detection where we mostly have examples of “typical” behavior of ordinary users. The set of possible intrusions is diverse and generating a representative set of negative examples could be challenging. A small set of negative examples (intrusions), those we already know about, might adversely bias our detector towards identifying the type of intrusions we have already taken steps to avoid. Similarly, in image or document retrieval, user might provide example images specifying what they want, but not any representative set of negative examples. The problem in this case is to rank the remaining images in the database according to how well they belong to the set that the user provided.

If we had access to the full distribution over positive examples, we could try to find the minimum volume set that contains at least $1 - \nu$ fraction of examples. The idea would be to carve out a small set that contains most positive examples, the “typical” examples. The parameter ν in this case would specify the false negative rate, i.e., positive examples that we will erroneously call negative. In practice, however, we only have access to a finite number of positively labeled points. We will formulate a 1-class SVM for best enclosing the available points within some parametric class of separating boundaries.

In the simplest formulation, we find a linear classifier such that $\underline{\theta} \cdot \underline{x} \geq \rho$ for all the positive examples $\underline{x}_1, \dots, \underline{x}_n$. We will try to find the largest ρ , while limiting the norm $\|\underline{\theta}\|$, so as to get the “tightest fit”. Given a training set of positive points $\underline{x}_1, \dots, \underline{x}_n$, we will formulate the estimation problem as

$$\min \frac{1}{2} \|\underline{\theta}\|^2 - \rho \quad \text{subject to } \underline{\theta} \cdot \underline{x}_i \geq \rho \quad \text{for all } i = 1, \dots, n \quad (1)$$

Note that we could have simply fixed the value of $\rho > 0$. In either case, we get a maximum margin separator for the positive points from origin with margin $\hat{\rho}/\|\hat{\underline{\theta}}\|$. However, we prefer the above formulation as it can be extended more easily to omit a certain fraction of points (see below). The optimization problem also doesn’t break if we cannot separate the points from origin (setting $\underline{\theta} = \underline{0}$ and $\rho = 0$ is always feasible). Note, however, that ρ will never be negative. Figure 1 below illustrates the separating boundary and the corresponding “support vectors”, i.e, points for which the inequalities are tight.

The figure looks a bit unsatisfying since it is hard to see that the linear boundary “encloses”

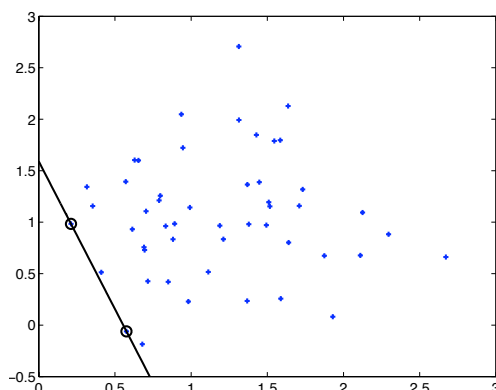


Figure 1: 1-class linear SVM. The decision boundary specifies \underline{x} such that $\hat{\underline{\theta}} \cdot \underline{x} - \hat{\rho} = 0$.

the set of points. We can remedy this with kernels but, before doing so, need to derive the dual problem for Eq.(1).

1-class dual. The dual formulation can be derived as before. The Lagrangian in this case is

$$\mathcal{L}(\underline{\theta}, \rho, \alpha) = \frac{1}{2} \|\underline{\theta}\|^2 - \rho - \sum_{i=1}^n \alpha_i [\underline{\theta} \cdot \underline{x}_i - \rho] \quad (2)$$

where $\alpha_i \geq 0$ are the Lagrange multipliers associated with the inequality constraints. By taking derivatives with respect to $\underline{\theta}$ and ρ , we get

$$\frac{\partial}{\partial \underline{\theta}} \mathcal{L}(\underline{\theta}, \rho, \alpha) = \underline{\theta} - \sum_{i=1}^n \alpha_i \underline{x}_i = 0 \quad (3)$$

$$\frac{\partial}{\partial \rho} \mathcal{L}(\underline{\theta}, \rho, \alpha) = -1 + \sum_{i=1}^n \alpha_i = 0 \quad (4)$$

If we now substitute the resulting $\underline{\theta}(\alpha)$ back into the Lagrangian, and incorporate the new constraint $\sum_{i=1}^n \alpha_i = 1$, we obtain the following dual problem:

$$\max \quad -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j (\underline{x}_i \cdot \underline{x}_j) \quad \text{subject to} \quad \alpha_i \geq 0, \quad \sum_{i=1}^n \alpha_i = 1 \quad (5)$$

where $(\underline{x}_i \cdot \underline{x}_j)$ can be replaced by any kernel function such as the radial basis kernel. Once we solve the quadratic programming problem, we can reconstruct the positive set as all \underline{x}

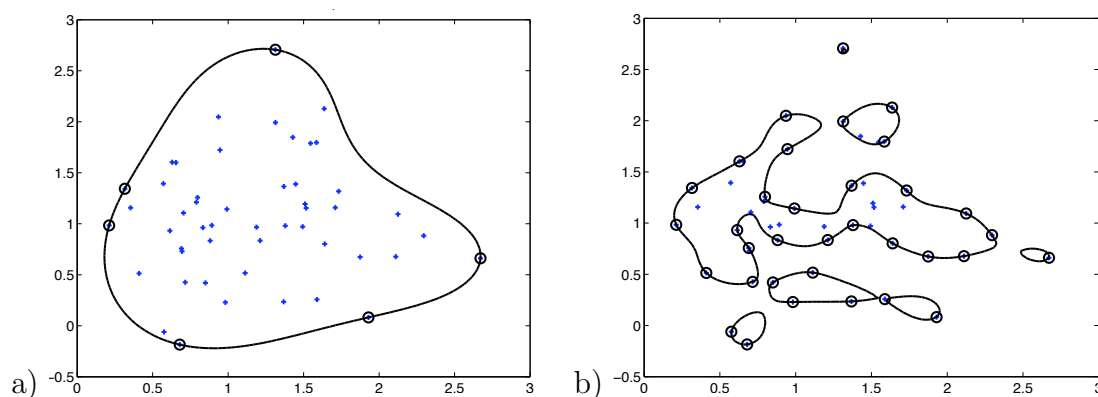


Figure 2: a) 1-class radial basis SVM. The decision boundary specifies \underline{x} such that $\sum_{i=1}^n \hat{\alpha}_i K(\underline{x}_i, \underline{x}) - \hat{\rho} = 0$. b) The same figure but with somewhat smaller kernel width parameter σ .

that satisfy

$$\underline{\theta}(\hat{\alpha}) \cdot \underline{x} - \hat{\rho} = \sum_{i=1}^n \hat{\alpha}_i (\underline{x}_i \cdot \underline{x}) - \hat{\rho} \geq 0 \quad (6)$$

For any given $\underline{\theta}(\hat{\alpha})$, the corresponding $\hat{\rho}$ will always take the largest value that does not violate the constraints. In other words,

$$\hat{\rho} = \min_{j=1, \dots, n} \{ \underline{\theta}(\hat{\alpha}) \cdot \underline{x}_j \} = \min_{j=1, \dots, n} \left\{ \sum_{i=1}^n \hat{\alpha}_i (\underline{x}_i \cdot \underline{x}_j) \right\} \quad (7)$$

Figure 2a) shows the same example as before, now with the radial basis kernel. The result might not look as good if we selected too small a value for the width parameter σ in the radial basis kernel $K(\underline{x}, \underline{x}') = \exp(-(1/2)\|\underline{x} - \underline{x}'\|^2/\sigma^2)$. Small σ might create negatively labeled islands inside the cloud of points. See Figure 2b) for an example.

Omit ν -fraction of points. The positive examples themselves may contain outliers that we should omit in finding a separator for the “typical” set of positive examples. Can we change the previous method to omit exactly a ν -fraction of points? We can almost get this by introducing slack variables for the inequality constraints with an appropriate penalty. Specifically, the primal formulation is given by

$$\min \frac{1}{2} \|\underline{\theta}\|^2 - \rho + \frac{1}{\nu n} \sum_{i=1}^n \xi_i \quad \text{subject to} \quad \underline{\theta} \cdot \underline{x}_i \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \text{for all } i \quad (8)$$

So, exactly what can we say about the solution to this problem? We can show that a) ν is a *lower* bound on the fraction of support vectors (tight inequalities), and b) ν is an *upper* bound on the fraction of points with non-zero slack. The interesting bit is that we would not get this desirable property if we fixed the value of ρ . Let's understand this a bit further. We will only show part b).

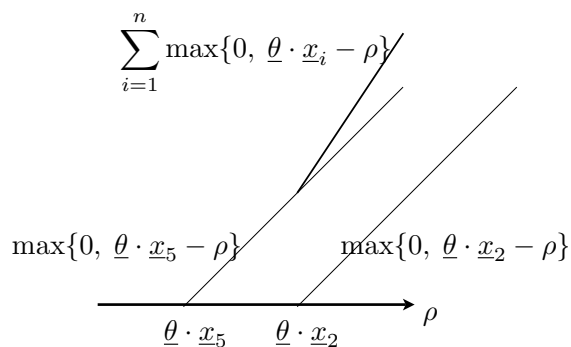
We can re-write the above optimization problem in terms of $\underline{\theta}$ and ρ alone by setting the slack variables to their optimal values for any fixed $\underline{\theta}$ and ρ . Since their values are penalized, they will take the smallest non-negative values that satisfy the inequalities. The inequalities can be written as $\xi_i \geq \rho - \underline{\theta} \cdot \underline{x}_i$ so that either $\xi_i = 0$ (when the right hand side is negative) or $\xi_i = \rho - \underline{\theta} \cdot \underline{x}_i$. More compactly,

$$\xi_i(\underline{\theta}, \rho) = \max\{0, \rho - \underline{\theta} \cdot \underline{x}_i\} \quad (9)$$

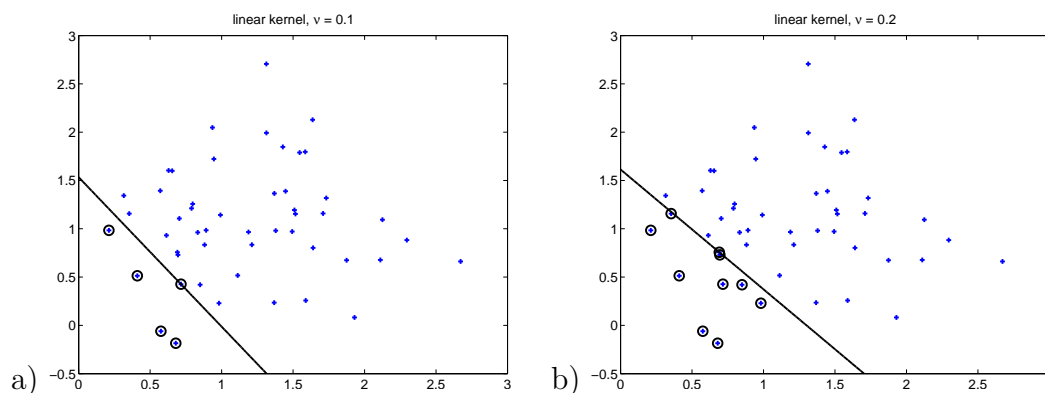
As a result, the optimization problem in terms $\underline{\theta}$ and ρ is given by

$$\min \frac{1}{2} \|\underline{\theta}\|^2 - \rho + \frac{1}{\nu n} \sum_{i=1}^n \max\{0, \rho - \underline{\theta} \cdot \underline{x}_i\} \quad (10)$$

Note that there are no more constraints since ξ 's were set to satisfy them. Let's keep $\underline{\theta}$ fixed and see what happens when we increase ρ starting from some negative value. Each term $\max\{0, \rho - \underline{\theta} \cdot \underline{x}_i\}$ remains zero (zero slack) until $\rho = \underline{\theta} \cdot \underline{x}_i$ after which it will increase linearly with ρ (slack increases). We select ρ such that the reduction in $-\rho$ is balanced with the increasing slacks. Since the slacks are penalized by $1/(\nu n)$, we can have a balance only if at most νn terms $\max\{0, \rho - \underline{\theta} \cdot \underline{x}_i\}$ increase with ρ (slacks are non-zero). The figure below illustrates this further.



The dual problem is modified only slightly with the introduction of slack variables (we omit

Figure 3: 1-class linear SVM with a) $\nu = 0.1$ and b) $\nu = 0.2$.

the derivation):

$$\max -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j (\underline{x}_i \cdot \underline{x}_j) \quad \text{subject to} \quad 0 \leq \alpha_i \leq 1/(\nu n), \quad \sum_{i=1}^n \alpha_i = 1 \quad (11)$$

How should we determine $\hat{\rho}$ now? There are two ways to think about this. We could simply follow the above description and find the largest ρ subject to

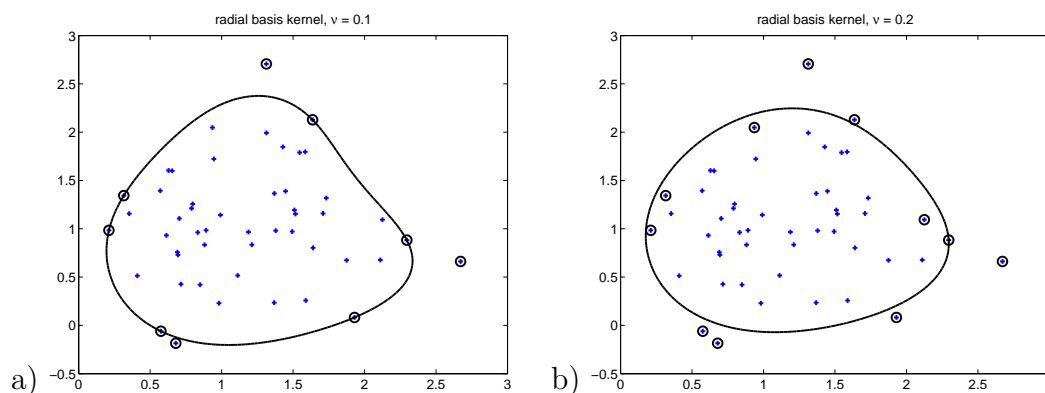
$$\sum_{j=1}^n \max \{0, \underline{\theta}(\hat{\alpha}) \cdot \underline{x}_j - \rho\} = \sum_{j=1}^n \max \left\{ 0, \sum_{i=1}^n \hat{\alpha}_i (\underline{x}_i \cdot \underline{x}_j) - \rho \right\} \leq \nu n \quad (12)$$

Alternatively, we can use the properties of the solution with Lagrange multipliers (complementary slackness). Accordingly, if $\alpha_i \in (0, 1/(\nu n))$, then $\underline{\theta}(\hat{\alpha}) \cdot \underline{x}_i = \rho$. In other words, if $\alpha_i = 0$ then we could have $\underline{\theta}(\hat{\alpha}) \cdot \underline{x}_i > \rho$. Similarly, if $\alpha_i = 1/(\nu n)$ then we might have $\underline{\theta}(\hat{\alpha}) \cdot \underline{x}_i < \rho$ (otherwise not). The slack can be non-zero only if $\alpha_i = 1/(\nu n)$. These constraints arise in the course of deriving the dual and are not therefore immediate from the equations we have here.

The figures below illustrate the linear and radial basis kernel solutions for different values of ν .

Minimum enclosing ball. We can also change the formulation slightly and use $\underline{\theta}$ to represent the center of a ball that encloses the points. We find the center that minimizes the radius of the enclosing ball. The optimization problem for this is given by

$$\min R^2 \quad \text{subject to} \quad \|\underline{\theta} - \underline{x}_i\|^2 \leq R^2, \quad i = 1, \dots, n \quad (13)$$

Figure 4: 1-class radial basis SVM with a) $\nu = 0.1$ and b) $\nu = 0.2$.

The “support vectors” in this case correspond to points that are right at the boundary of the enclosing ball, i.e., for which the inequalities are tight. As before, we can extend the formulation to omit a ν -fraction of the points:

$$\min R^2 + \frac{1}{\nu n} \sum_{i=1}^n \xi_i \quad \text{subject to} \quad \|\underline{\theta} - \underline{x}_i\|^2 \leq R^2 + \xi_i, \quad i = 1, \dots, n \quad (14)$$

This is actually the same problem as before if the examples have the same norm, i.e., if $\|\underline{x}_i\| = 1$ for all i . In general, however, the result is different. The corresponding dual problem is (derivation omitted)

$$\max - \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j (\underline{x}_i \cdot \underline{x}_j) + \sum_{i=1}^n \alpha_i (\underline{x}_i \cdot \underline{x}_i) \quad \text{subject to} \quad 0 \leq \alpha_i \leq 1/(\nu n), \quad \sum_{i=1}^n \alpha_i = 1 \quad (15)$$

(note the reduction to the previous formulation if $(\underline{x}_i \cdot \underline{x}_i) = \|\underline{x}_i\|^2 = 1$). The correspondence between the primal and the dual is given by

$$\theta(\alpha) = \sum_{i=1}^n \alpha_i \underline{x}_i \quad (16)$$

In other words, the “center” we find will be a weighted average of the “support vectors”.

Finding the value of \hat{R}^2 is analogous to ρ before, i.e., we can evaluate it on the basis of points for which $\alpha_i \in (0, 1/(\nu n))$. Once we have obtained \hat{R}^2 and $\hat{\alpha}$, we would say that a

new point \underline{x} is “typical” if

$$\|\theta(\hat{\alpha}) - \underline{x}\|^2 \leq R^2 \quad \text{or} \quad \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j (\underline{x}_i \cdot \underline{x}_j) - 2 \sum_{i=1}^n \alpha_i (\underline{x}_i \cdot \underline{x}) + (\underline{x} \cdot \underline{x}) \leq R^2 \quad (17)$$

Multi-class classification

Most classification problems in practice involve more than two classes: identifying a person in an image, predicting phonemes from speech, associating a gene with biological processes, or predicting cancer type from cell samples. We have so far focused only on binary (two class) classification tasks. However, methods we have learned, including perceptron and SVMs, can be used as subroutines in solving multi-class classification tasks. Indeed, we can reduce any multi-class classification problem to a set of binary classification tasks through what are known as output codes.

Consider a simple three class classification task where $y \in \{1, 2, 3\}$. Perhaps the most common output code is “one versus all” where the corresponding binary classification tasks are obtained by highlighting examples of each particular class as positive (+1) and treating the remaining examples as negative (−1). So, in this case, any k -class classification task will be turned into k binary classification tasks. When $k = 3$, we obtain the following output code matrix R :

$$\begin{array}{l} \\ \\ \\ \end{array} \begin{array}{ccc} \text{task 1} & \text{task 2} & \text{task 3} \\ \left[\begin{array}{ccc} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \end{array} \right] & = R & \end{array} \quad (18)$$

Each row corresponds to one of the original labels $y = 1, 2, 3$ and the columns define how the labels are translated into binary labels in each component task. So, for example, in solving task 3, any example labeled $y = 2$ has a target binary target label $R(2, 3) = -1$.

Once we have the output code in place, we can *separately* train classifiers $f_1(\underline{x})$, $f_2(\underline{x})$, $f_3(\underline{x})$ to solve each associated binary task. For example, if $(\underline{x}_1, y_1), \dots, (\underline{x}_n, y_n)$ represents the original labeled 3-class training set, then $f_1(\underline{x})$ is trained with $(\underline{x}_1, y_1^{(1)}), \dots, (\underline{x}_n, y_n^{(1)})$ where $y_i^{(1)} = R(y_i, 1)$. That is, $y_i^{(1)}$ corresponds to the first column of R .

There are many possible output codes. For example, we could use a pairwise output code

defined as

$$R = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{bmatrix} \quad (19)$$

where the binary task 1 distinguishes examples labeled 1 and 2 without considering examples from class 3. Similarly, task 2 separates 1 from 3 without examples from class 2, and so on. We use 0 in the output code to indicate that those examples are not part of the corresponding binary task. For instance, examples corresponding to class 3 would be excluded from the training set for the first binary classifier.

We need at least $\log_2(k)$ binary tasks to solve any k class classification task so R will have to have at least that many columns. So, for $k = 3$, we could use a minimal output code matrix (not unique)

$$R = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \quad (20)$$

Why not use such a minimal code? Will answer this shortly.

The last step in our approach is to combine the outputs of trained binary classifiers $\hat{f}_1(\underline{x})$, $\hat{f}_2(\underline{x})$, and $\hat{f}_3(\underline{x})$ into a full three-way classifier. Clearly, prediction of original labels has to be based on the output code R . If $f_i(\underline{x}) \in \{-1, 1\}$, we can simply predict the class label that best agrees with the binary predictions:

$$\hat{y} = \operatorname{argmax}_{y \in \{1,2,3\}} \left\{ \sum_{i=1}^m R(y, i) \hat{f}_i(\underline{x}) \right\} \quad (21)$$

where $m = 3$ is the number of binary tasks. If each component classifier $\hat{f}_i(\underline{x})$ predicts $R(y, i)$ consistent with the true label y , then $\sum_{i=1}^m R(y, i) \hat{f}_i(\underline{x})$ will be maximized for that y . The classifier outputs may be contradictory, of course (the binary classifiers are not perfect).

Using $f_i(\underline{x}) \in \{-1, 1\}$ omits how strongly each classifier insists on its binary label. We could instead use the discriminant function values. In other words, for linear classifiers, we could use $\hat{f}_i(\underline{x}) = \hat{\theta}^{(i)} \cdot \underline{x} + \hat{\theta}_0^{(i)}$ in Eq.(21) and this is often advantageous.

A bit more general approach to deriving predictions from output codes comes from specifying a $\text{Loss}(y_b, f_i(\underline{x}))$ that measures how poorly our discriminant function matches a

particular binary label y_b . In this case, we would predict the class that is most consistent with the classifiers (minimum loss):

$$\hat{y} = \operatorname{argmin}_{y \in \{1,2,3\}} \left\{ \sum_{i=1}^m \operatorname{Loss}(R(y, i), \hat{f}_i(\underline{x})) \right\} \quad (22)$$

There are many possible losses we could define. For example, we could use the Hinge loss similarly to SVMs:

$$\operatorname{Loss}(R(y, i), \hat{f}_i(\underline{x})) = \max \left\{ 0, 1 - R(y, i) \hat{f}_i(\underline{x}) \right\} = \left(1 - R(y, i) \hat{f}_i(\underline{x}) \right)^+ \quad (23)$$

Note that the loss is not zero when $R(y, i) = 0$; otherwise labels with more zeros in the corresponding rows would be preferred.

So, back to our question. Which output code to use? Note that some of the binary classification tasks corresponding to a particular output code R may be harder to solve than others. Harder tasks lead to poorly performing binary classifiers. However, a poor performance on one task does not affect how we train the classifiers for the other binary tasks. This is suboptimal since the predicted class label may, as a result, rely heavily on the poorly performing classifier. This is indeed why the minimal output code is almost never ideal (a single poorly performing component classifier will degrade the performance considerably). At the other extreme, we can concatenate multiple output codes as in

$$R = \begin{bmatrix} 1 & 1 & 0 & +1 & -1 & -1 \\ -1 & 0 & 1 & -1 & +1 & -1 \\ 0 & -1 & -1 & -1 & -1 & +1 \end{bmatrix} \quad (24)$$

While there is a computational cost of using more (compensating) binary tasks, such codes work quite well in practice.

We can think of the output codes in terms of communication. For each input example \underline{x} , we have to send the corresponding label symbol $y \in \{1, 2, 3\}$ through a communication channel. The symbol is first encoded into a bit vector $\underline{R}(y) = (R(y, 1), \dots, R(y, m))$, where m is the length of the output code, and then sent through an uncertain channel. We obtain a possibly perturbed version of the bits $\underline{f}(\underline{x}) = (\hat{f}_1(\underline{x}), \dots, \hat{f}_m(\underline{x}))$ (with real valued components), and need to reconstruct the original symbol. If the errors generated by each classifier can be modeled as independent noisy flips of the target binary labels, then the setting reduces to a channel coding problem in information theory. However, our setting is a bit more complicated as there is a trade-off between two key things. On one hand, we would like good error correcting properties for the output code so that the original label

symbol can be reconstructed from a noisy version of received (predicted) binary labels. On the other hand, we need to keep the binary tasks for the component classifiers $f_i(\underline{x})$ easily solvable so that they generalize well. The balance between error correction and difficulty of the tasks is an interesting one. Let's explore this a bit further.

We will first need to quantify how well a particular output code can correct errors in the binary classifiers. The key measure here is the distance between any two rows in the output code, i.e., pairs of (row) vectors $\underline{R}(y)$ and $\underline{R}(y')$ for $y \neq y'$. We will measure this distance via a (generalized) Hamming distance:

$$\Delta(\underline{R}(y), \underline{R}(y')) = \sum_{i=1}^m \frac{(1 - R(y, i)R(y', i))}{2} \quad (25)$$

When the output code only contains ± 1 entries, this simply counts the number of different elements. The larger the minimum distance $\rho = \min_{y \neq y'} \Delta(\underline{R}(y), \underline{R}(y'))$, the more errors we can correct. Roughly speaking, if the rows are ρ apart, then we can withstand one less than $\rho/2$ errors in the binary classifiers. $\rho = 2$ for the one-versus-all code. Indeed, even a single error in the binary predictions would create a tie between two multi-class labels. $\rho = 1$ for the minimal code; a single binary error would cause us to confidently select a wrong multi-class label.

We can turn this around and ask about the losses we would have to incur in the binary predictions if we made a single multi-class error. In other words, we would expect that, when ρ is large (good error correction), whenever we make a multi-class error, we have to make many binary errors. Indeed, for the Hinge loss discussed above, we can show that¹

$$\sum_{i=1}^m \text{Loss}(R(y, i), f_i(\underline{x})) \geq \rho \quad (26)$$

if we make a multi-class error on (\underline{x}, y) . This allows us to bound the multi-class error on the training set by

$$\text{multi-class error} \leq \frac{1}{\rho} \sum_{t=1}^n \sum_{i=1}^m \text{Loss}(R(y_t, i), f_i(\underline{x}_t)) \quad (27)$$

since each multi-class error corresponds to binary losses greater than ρ . This is the result we were looking for in terms of quantifying the balance between error correction and the difficulty of the binary tasks. If we can solve the binary tasks well (small loss), and the

¹See Allwein et al., "Reducing multi-class to binary", Journal of Machine Learning Research, 2000.

output code has good error correction properties (large ρ), the resulting multi-class error will be small. An output code with large ρ may, however, specify very difficult binary tasks and therefore cause the losses to be high.

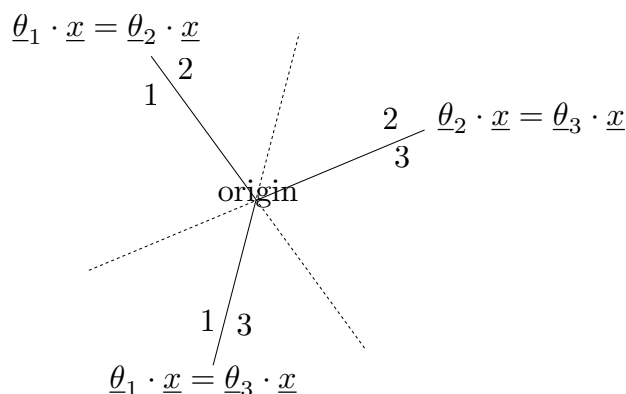
We finally note that SVMs correspond to minimizing the Hinge loss over the training set. Thus, if we solve each binary task using SVMs, we are explicitly trying to minimize the right hand side in the above equation. The better we can solve the binary tasks, the smaller the losses, and the smaller the multi-class error.

Multi-class SVM. We have so far trained the component classifiers independently from each other. This is a strength and a weakness. The weakness is that a poor performance in one task is not immediately compensated by the other tasks. The only remedy we have is to select another output code that either avoids or appropriately complements the difficult task.

We can instead try to set up a joint optimization problem for the component classifiers so as to train a multi-class classifier. Let's use as components simple linear classifiers through origin, i.e., $f_i(\underline{x}) = \underline{\theta}_i \cdot \underline{x}$. We will then require that the classifier corresponding to the correct label wins in the sense that its discriminant function value is the highest with a little room to spare. The problem to solve is then

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{y=1}^k \|\underline{\theta}_y\|^2 \quad \text{subject to} \\ & (\underline{\theta}_{y_i} \cdot \underline{x}_i) \geq (\underline{\theta}_y \cdot \underline{x}_i) + 1 \quad \text{for all } y \neq y_i, \quad i = 1, \dots, n \end{aligned} \quad (28)$$

Note that this may look like the one-versus-all output code but there are two key differences. First, the component classifiers are trained jointly so they will compensate each other's errors. Second, we predict (decode) based on how much each component classifier supports the corresponding label: $\hat{y} = \arg \max_y \{\underline{\theta}_y \cdot \underline{x}\}$. The resulting decision regions have a simple geometric interpretation. We can identify them by combining pairwise regions, i.e., sets of examples where we would prefer one class over another. These are given simply by lines (hyper-planes) through origin: $\underline{\theta}_y \cdot \underline{x} = \underline{\theta}_{y'} \cdot \underline{x}$ for $y, y' \in \{1, 2, 3\}$, as in the figure below. The dotted lines indicate how these pairwise separator hyper-planes extend into regions dominated by other labels. The decision regions are carved out by the solid lines.



Before proceeding to solve this optimization problem in the dual, it will be helpful (for later utility) to write the problem in a bit more general form. To this end, we define

$$\underline{\theta} = \begin{bmatrix} \underline{\theta}_1 \\ \dots \\ \underline{\theta}_y \\ \dots \\ \underline{\theta}_k \end{bmatrix}, \quad \phi(\underline{x}, y) = \begin{bmatrix} \underline{0} \\ \dots \\ \underline{x} \\ \dots \\ \underline{0} \end{bmatrix} \quad (29)$$

where the location of \underline{x} in $\phi(\underline{x}, y)$ guarantees that $\underline{\theta} \cdot \phi(\underline{x}, y) = \underline{\theta}_y \cdot \underline{x}$. We also define $\bar{\Delta}(y_i, y) = 1 - \delta(y_i, y)$ so that it is one if $y_i \neq y$ and zero otherwise. As a result, the primal version of the optimization problem can be re-written as

$$\begin{aligned} \min \quad & \frac{1}{2} \|\underline{\theta}\|^2 \quad \text{subject to} \\ & \underline{\theta} \cdot \phi(\underline{x}_i, y_i) \geq \underline{\theta} \cdot \phi(\underline{x}_i, y) + \bar{\Delta}(y_i, y) \quad \text{for all } y = 1, \dots, k, \quad i = 1, \dots, n \end{aligned} \quad (30)$$

This form will be useful later on when y itself has structure. For example, y might correspond to an annotation of the input sequence. We will cover structured prediction later in the course.

Now, we proceed to get the dual problem as with binary SVMs. We will introduce Lagrange multipliers $\alpha_{y;i} \geq 0$ for the inequality constraints, obtain the Lagrangian, find the minimizing $\underline{\theta}$ as a function of the Lagrange multipliers, and substitute this $\hat{\underline{\theta}}(\alpha)$ back into

the Lagrangian to get the dual. To this end,

$$\mathcal{L}(\underline{\theta}, \alpha) = \frac{1}{2} \|\underline{\theta}\|^2 - \sum_{i=1}^n \sum_{y=1}^k \alpha_{y;i} [\underline{\theta} \cdot \phi(\underline{x}_i, y_i) - \underline{\theta} \cdot \phi(\underline{x}_i, y) - \bar{\Delta}(y_i, y)] \quad (31)$$

$$\frac{\partial}{\partial \underline{\theta}} \mathcal{L}(\underline{\theta}, \alpha) = \underline{\theta} - \sum_{i=1}^n \sum_{y=1}^k \alpha_{y;i} [\phi(\underline{x}_i, y_i) - \phi(\underline{x}_i, y)] = 0 \quad (32)$$

$$\Rightarrow \underline{\theta}(\alpha) = \sum_{i=1}^n \sum_{y=1}^k \alpha_{y;i} [\phi(\underline{x}_i, y_i) - \phi(\underline{x}_i, y)] \quad (33)$$

If we now substitute $\underline{\theta}(\alpha)$ back into the Lagrangian and rearrange terms, we get

$$\begin{aligned} \mathcal{L}(\underline{\theta}(\alpha), \alpha) &= \sum_{i=1}^n \sum_{y=1}^k \alpha_{y;i} \bar{\Delta}(y_i, y) - \\ &\quad \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \sum_{y=1}^k \sum_{y'=1}^k \alpha_{y;i} \alpha_{y';j} [\phi(\underline{x}_i, y_i) - \phi(\underline{x}_i, y)] \cdot [\phi(\underline{x}_j, y_j) - \phi(\underline{x}_j, y')] \end{aligned} \quad (34)$$

which is to be maximized subject to $\alpha_{y;i} \geq 0$ for all $y = 1, \dots, k$, $i = 1, \dots, n$. This can be simplified further by noting that, in our setting, $\phi(\underline{x}, y) \cdot \phi(\underline{x}', y') = (\underline{x} \cdot \underline{x}') \delta(y, y')$ where $\delta(y, y') = 1$ if $y = y'$ and zero otherwise.

Once we have found the maximizing $\hat{\alpha}$, we can predict the label for a new example \underline{x} according to

$$\hat{y} = \arg \max_y \{ \underline{\theta}(\hat{\alpha}) \cdot \phi(\underline{x}, y) \} \quad (35)$$

$$= \arg \max_y \left\{ \sum_{i=1}^n \sum_{y'=1}^k \hat{\alpha}_{y';i} [\phi(\underline{x}_i, y_i) - \phi(\underline{x}_i, y')] \cdot \phi(\underline{x}, y) \right\} \quad (36)$$

which again can be simplified further.