Lecture 15

Naive Bayes, neighbors, and learned distances

Two ways to classify an observation:

Model its features within each class, or look at nearby observations.

Then ask: can we learn what “nearby” should mean?

Our data and prediction task

$$X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$$

$$\mathbf{x}_i = (x_{1i}, \ldots, x_{di})^\mathsf{T}, \qquad y_i \in \{1, \ldots, c\}$$

Each column is one observation. Each row is one feature.

Learn from the training observations, then predict a new label.

Bayes classification

$$P(y = r \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y = r)\, \pi_r}{p(\mathbf{x})}$$

$$\pi_r = P(y = r), \qquad p(\mathbf{x}) = \sum_{s=1}^{c} p(\mathbf{x} \mid y = s)\pi_s$$

For a fixed observation, the denominator is the same for every class.

$$h(\mathbf{x}) = \arg\max_r P(y = r \mid \mathbf{x}) = \arg\max_r p(\mathbf{x} \mid y = r)\pi_r$$

Naive Bayes: a simplifying assumption

Estimating a joint density can require many parameters.

Naive Bayes assumes that features are independent given the class.

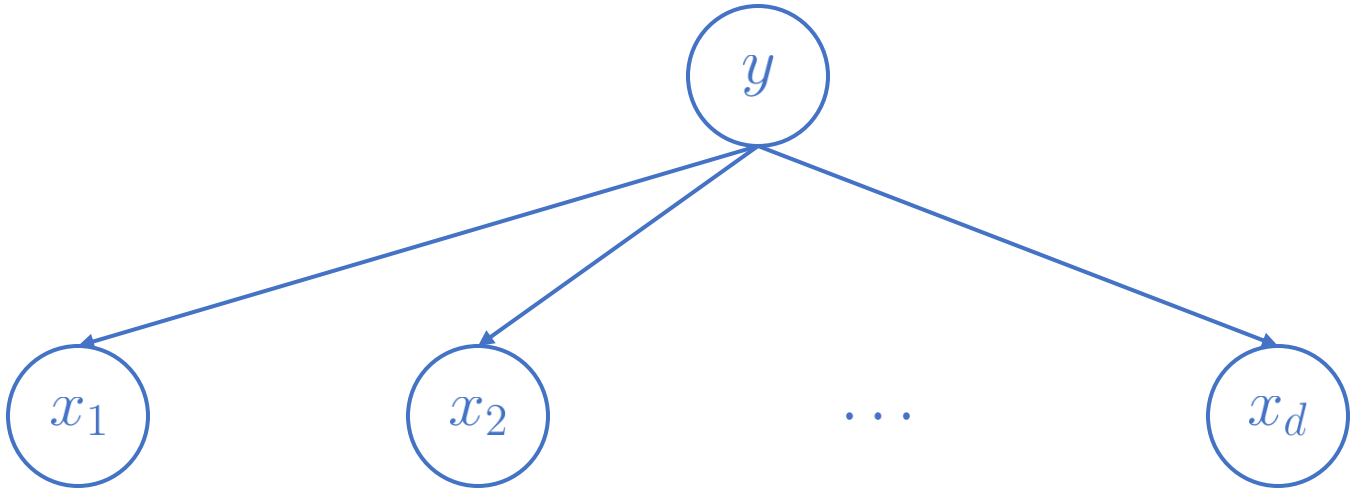
This lets us estimate one feature distribution at a time, within each class.

Conditional independence

$$p(\mathbf{x} \mid y = r) = \prod_{j=1}^{d} p(x_j \mid y = r)$$

Given the class, the model treats the features independently.

Features can still be dependent when the classes are mixed together.



The Naive Bayes decision rule

$$h(\mathbf{x}) = \arg\max_r \ \pi_r \prod_{j=1}^{d} p(x_j \mid y = r)$$

Taking a logarithm preserves the maximizing class.

$$g_r(\mathbf{x}) = \log \pi_r + \sum_{j=1}^{d} \log p(x_j \mid y = r)$$

$$h(\mathbf{x}) = \arg\max_r \ g_r(\mathbf{x})$$

Binary features: the Bernoulli model

$$x_j \in \{0, 1\}, \qquad p_{jr} = P(x_j = 1 \mid y = r)$$

$$p(x_j \mid y = r) = p_{jr}^{x_j}(1 - p_{jr})^{1-x_j}$$

The exponent selects the probability for the observed feature value.

$$g_r(\mathbf{x}) = \log \pi_r + \sum_j [x_j \log p_{jr} + (1 - x_j) \log(1 - p_{jr})]$$

Estimating the probabilities

$$n_r = \sum_{i=1}^{n} \mathbf{1}\{y_i = r\}, \qquad \widehat{\pi}_r = \frac{n_r}{n}$$

Count how often the feature is present among observations in this class.

$$n_{jr} = \sum_{i:y_i=r} x_{ji}, \qquad \widehat{p}_{jr} = \frac{n_{jr}}{n_r}$$

If the feature never appears in this class, this estimate is zero.

Avoiding zero probabilities

Use a small positive value when a fitted probability is exactly zero or one.

$$\widehat{p}_{jr} = 0 \;\Rightarrow\; \widehat{p}_{jr} \leftarrow \varepsilon, \qquad \widehat{p}_{jr} = 1 \;\Rightarrow\; \widehat{p}_{jr} \leftarrow 1 - \varepsilon$$

$$\widehat{P}(x_j = 0 \mid y = r) = 1 - \widehat{p}_{jr}, \qquad 0 < \varepsilon < \tfrac{1}{2}$$

Unseen feature, $\varepsilon = 0.001$: $\widehat{p}_{jr} = 0.001, \quad 1 - \widehat{p}_{jr} = 0.999$

The score for a two-feature observation

$$s_r = p(\mathbf{x} \mid y = r)\pi_r$$

$$= p(x_1, x_2 \mid y = r)\pi_r$$

$$= p(x_1 \mid y = r)p(x_2 \mid y = r)\pi_r$$

$$\mathbf{x} = (1, 0)^\mathsf{T} \quad \Longrightarrow \quad s_r = p_{1r}(1 - p_{2r})\pi_r$$

A small prediction example

Two binary features: the email contains “offer” and contains “meeting”.

$\pi_1 = 0.4, \quad (p_{11}, p_{21}) = (0.8, 0.1)$ for spam

$\pi_0 = 0.6, \quad (p_{10}, p_{20}) = (0.2, 0.7)$ for non-spam

$\mathbf{x} = (1, 0)^\mathsf{T}$: “offer” appears and “meeting” is absent.

Computing the two class scores

$$s_1 = \pi_1 p_{11}(1 - p_{21}) = 0.4(0.8)(1 - 0.1) = 0.288$$

$$s_0 = \pi_0 p_{10}(1 - p_{20}) = 0.6(0.2)(1 - 0.7) = 0.036$$

The spam score is larger, so classify this email as spam.

To obtain posterior probabilities, divide each score by their sum.

Normalizing the class scores

$$P(y = 1 \mid \mathbf{x}) = \frac{s_1}{s_0 + s_1}$$

$$= \frac{0.288}{0.036 + 0.288} = \frac{8}{9} \approx 0.889$$

Predict spam because its posterior probability is larger.

The absent “meeting” feature also contributes evidence.

Comparing the log posterior odds

$$g(\mathbf{x}) = \log \frac{P(y=1 \mid \mathbf{x})}{P(y=0 \mid \mathbf{x})}$$

$$= \log \frac{\pi_1 \prod_j p(x_j \mid y=1)}{\pi_0 \prod_j p(x_j \mid y=0)}$$

$$= \log \frac{\pi_1}{\pi_0} + \sum_j \log \frac{p(x_j \mid y=1)}{p(x_j \mid y=0)}$$

$$= \log \frac{\pi_1}{\pi_0} + \sum_j \left[\log p(x_j \mid y=1) - \log p(x_j \mid y=0)\right]$$

Expanding each Bernoulli log probability

$$p(x_j \mid y = r) = p_{jr}^{x_j}(1 - p_{jr})^{1-x_j}$$

$$\log p(x_j \mid y = r) = x_j \log p_{jr} + (1 - x_j) \log(1 - p_{jr})$$

$$\log p(x_j \mid y = 1) - \log p(x_j \mid y = 0)$$

$$= x_j \left[\log p_{j1} - \log p_{j0}\right] + (1 - x_j) \left[\log(1 - p_{j1}) - \log(1 - p_{j0})\right]$$

Separating the constant and feature terms

$$g(\mathbf{x}) = \log \frac{\pi_1}{\pi_0} + \sum_j \left[x_j \log \frac{p_{j1}}{p_{j0}} + (1 - x_j) \log \frac{1 - p_{j1}}{1 - p_{j0}} \right]$$

$$= \log \frac{\pi_1}{\pi_0} + \sum_j \left[x_j \log \frac{p_{j1}}{p_{j0}} + \log \frac{1 - p_{j1}}{1 - p_{j0}} - x_j \log \frac{1 - p_{j1}}{1 - p_{j0}} \right]$$

$$= \underbrace{\log \frac{\pi_1}{\pi_0} + \sum_j \log \frac{1 - p_{j1}}{1 - p_{j0}}}_{\text{constant in } \mathbf{x}} + \sum_j x_j \underbrace{\left[\log \frac{p_{j1}}{p_{j0}} - \log \frac{1 - p_{j1}}{1 - p_{j0}} \right]}_{\text{coefficient of } x_j}$$

The linear log-odds score

$$b = \log \frac{\pi_1}{\pi_0} + \sum_j \log \frac{1 - p_{j1}}{1 - p_{j0}}$$

$$w_j = \log \frac{p_{j1}}{p_{j0}} - \log \frac{1 - p_{j1}}{1 - p_{j0}} = \log \frac{p_{j1}(1 - p_{j0})}{p_{j0}(1 - p_{j1})}$$

$$g(\mathbf{x}) = b + \sum_j w_j x_j = b + \mathbf{w}^\top \mathbf{x}$$

Predict class one when this log-odds score is positive.

Recovering the posterior probability

$$p = P(y = 1 \mid \mathbf{x}), \qquad 1 - p = P(y = 0 \mid \mathbf{x})$$

$$g(\mathbf{x}) = \log \frac{p}{1-p} \quad \Longrightarrow \quad -g(\mathbf{x}) = \log \frac{1-p}{p}$$

$$e^{-g(\mathbf{x})} = \frac{1-p}{p} \quad \Longrightarrow \quad 1 + e^{-g(\mathbf{x})} = \frac{p + 1 - p}{p} = \frac{1}{p}$$

$$P(y = 1 \mid \mathbf{x}) = p = \frac{1}{1 + e^{-g(\mathbf{x})}}$$

Continuous features: Gaussian Naive Bayes

$$x_j \mid y = r \sim \mathcal{N}(\mu_{jr}, \sigma^2_{jr})$$

$$p(x_j \mid y = r) = \frac{1}{\sqrt{2\pi\sigma^2_{jr}}} \exp\left[-\frac{(x_j - \mu_{jr})^2}{2\sigma^2_{jr}}\right]$$

Each feature has its own mean and variance within each class.

Estimate these quantities from the training observations in that class.

Taking the log of the Gaussian density

$$\log p(x_j \mid y = r) = \log \left\{ \frac{1}{\sqrt{2\pi\sigma_{jr}^2}} \exp\left[-\frac{(x_j - \mu_{jr})^2}{2\sigma_{jr}^2} \right] \right\}$$

$$= \log\left[(2\pi\sigma_{jr}^2)^{-1/2} \right] + \log\left[\exp\left(-\frac{(x_j - \mu_{jr})^2}{2\sigma_{jr}^2} \right) \right]$$

$$= -\frac{1}{2}\log(2\pi\sigma_{jr}^2) - \frac{(x_j - \mu_{jr})^2}{2\sigma_{jr}^2}$$

The Gaussian Naive Bayes class score

$$g_r(\mathbf{x}) = \log\left[\pi_r \prod_{j=1}^{d} p(x_j \mid y = r)\right] = \log \pi_r + \sum_{j=1}^{d} \log p(x_j \mid y = r)$$

$$= \log \pi_r + \sum_{j=1}^{d} \left[-\frac{1}{2}\log(2\pi\sigma_{jr}^2) - \frac{(x_j - \mu_{jr})^2}{2\sigma_{jr}^2}\right]$$

$$= \log \pi_r - \frac{1}{2}\sum_{j=1}^{d} \log(2\pi\sigma_{jr}^2) - \frac{1}{2}\sum_{j=1}^{d} \frac{(x_j - \mu_{jr})^2}{\sigma_{jr}^2}$$

This is a Gaussian classifier with diagonal class covariance matrices.

When the Gaussian boundary is linear

$$\sigma_{j1}^2 = \sigma_{j0}^2 = \sigma_j^2$$

$$g_1(\mathbf{x}) - g_0(\mathbf{x}) = \log \frac{\pi_1}{\pi_0} - \frac{1}{2} \sum_j \frac{(x_j - \mu_{j1})^2 - (x_j - \mu_{j0})^2}{\sigma_j^2}$$

$$(x_j - \mu_{j1})^2 - (x_j - \mu_{j0})^2 = -2x_j(\mu_{j1} - \mu_{j0}) + \mu_{j1}^2 - \mu_{j0}^2$$

$$g_1 - g_0 = \log \frac{\pi_1}{\pi_0} + \sum_j \frac{\mu_{j1} - \mu_{j0}}{\sigma_j^2} x_j - \frac{1}{2} \sum_j \frac{\mu_{j1}^2 - \mu_{j0}^2}{\sigma_j^2}$$

What Naive Bayes gains and loses

Few parameters and simple estimation make it a useful baseline.

Correlated features can count the same evidence more than once.

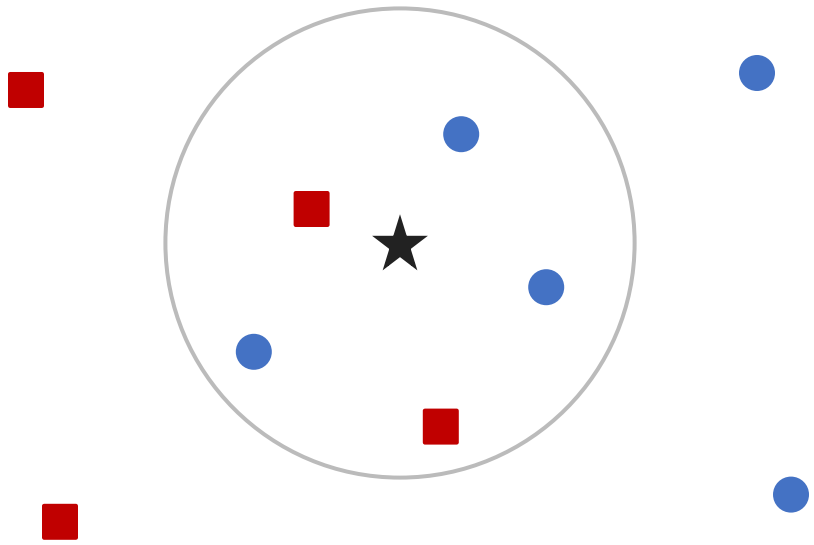
Good classification does not guarantee accurate posterior probabilities.

Next: estimate the class locally, using nearby training observations.

Nearest-neighbor classification

A query observation inherits the label of its nearest training observation.

For several neighbors, use the most common class.



Star: query

Five neighbors

Blue wins, 3 to 2

The local class proportions

$\mathcal{N}_k(\mathbf{x}) =$ indices of the k nearest training observations

$$\widehat{P}(y = r \mid \mathbf{x}) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x})} \mathbf{1}\{y_i = r\}$$

$$h(\mathbf{x}) = \arg\max_r \widehat{P}(y = r \mid \mathbf{x})$$

For five neighbors with labels blue, blue, red, blue, red, predict blue.

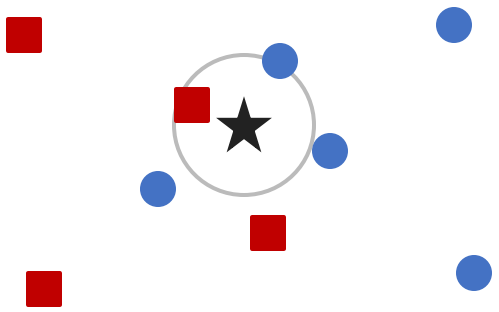
Choosing the neighborhood size

$k = 1$: very local decisions, sensitive to individual observations

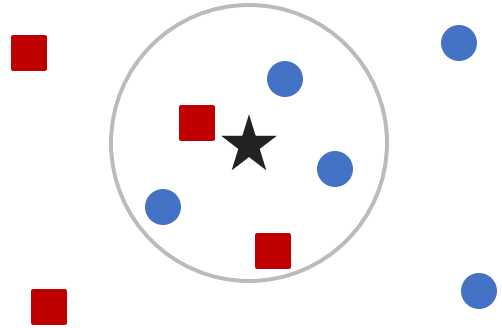
Larger k : more averaging, potentially more bias

If every training point is a neighbor, the majority class wins everywhere.

Select the neighborhood size using validation data.



One neighbor: red

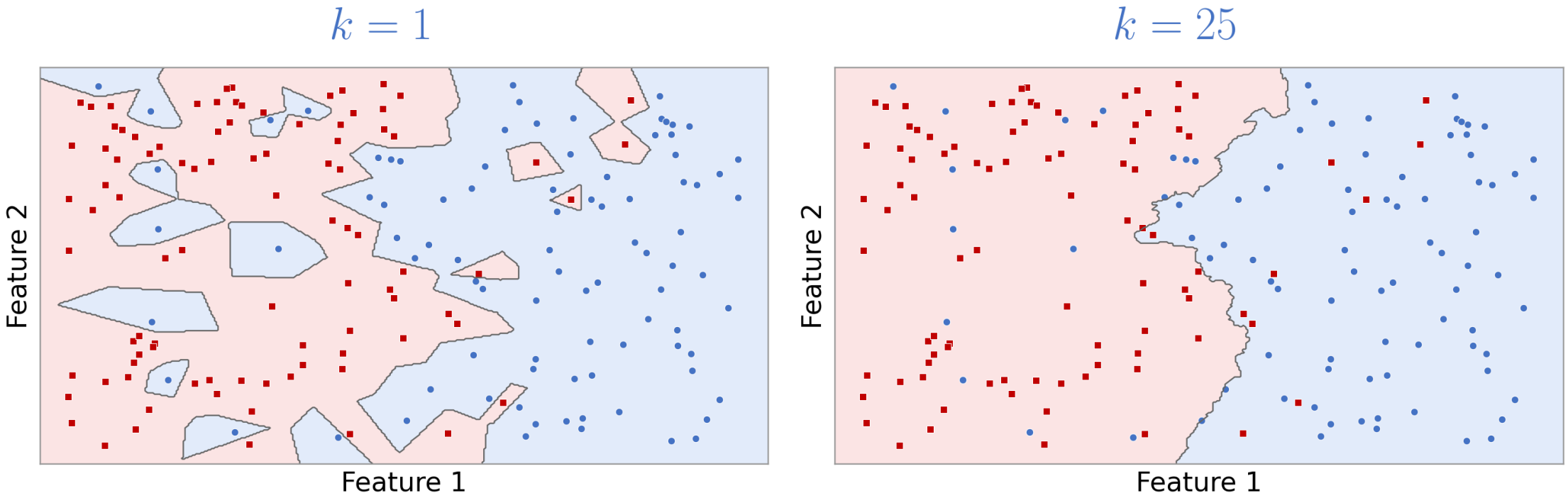


Five neighbors: blue

How the neighborhood size changes the boundary

The same training observations appear in both panels.

A larger neighborhood produces fewer small decision regions in this example.



Markers show training labels. Shading shows the predicted class.

Distance depends on feature scale

A feature measured in large numerical units can dominate distance.

$$\|\mathbf{x} - \mathbf{x}_i\|^2 = \sum_{j=1}^{d} (x_j - x_{ji})^2$$

$$\widetilde{x}_j = \frac{x_j - \widehat{\mu}_j}{\widehat{s}_j}$$

Fit scaling on the training data, then apply it to validation and test data.

Why irrelevant features hurt

$$\mathbf{x} = (\mathbf{x}_{\text{signal}}, \mathbf{x}_{\text{noise}})$$

$$\|\mathbf{x} - \mathbf{x}_i\|^2 = \|\mathbf{x}_{\text{signal}} - \mathbf{x}_{i,\text{signal}}\|^2 + \|\mathbf{x}_{\text{noise}} - \mathbf{x}_{i,\text{noise}}\|^2$$

Many irrelevant coordinates can dominate the useful differences.

As dimension grows, a fixed sample covers the space less densely.

The cost of a direct neighbor search

$$\|\mathbf{x} - \mathbf{x}_i\|^2 = \sum_{j=1}^{d} (x_j - x_{ji})^2$$

For one training observation, subtract and square each feature difference, then add.

d feature calculations $\implies$ $O(d)$ per training observation

n training observations $\implies$ $n \times O(d) = O(nd)$ per query

Can we learn a better distance?

Feature scaling helps, but it does not use the class labels.

A useful distance should emphasize differences relevant to the prediction task.

Can labels teach us which differences should matter?

Similarity depends on the task



Two faces may be similar in identity, glasses, or facial hair.

The same observations can therefore require different neighborhoods.

Supervision specifies which relationships the embedding should preserve.

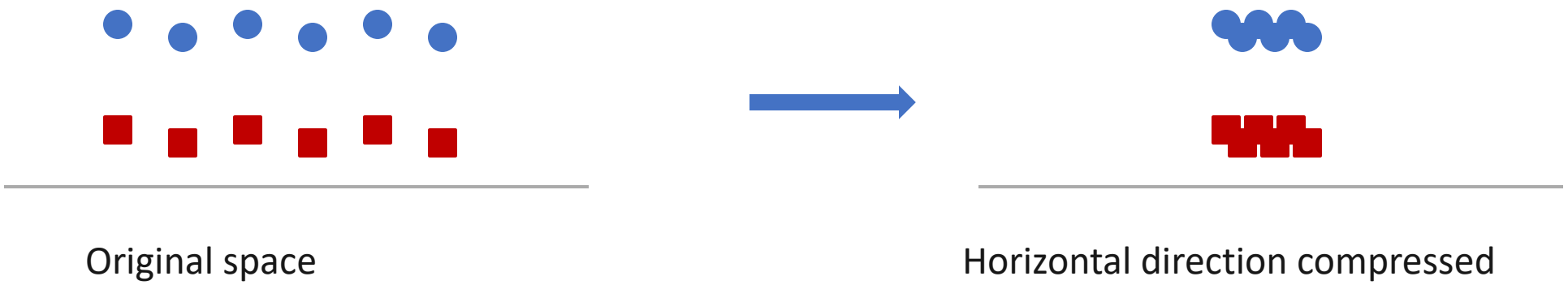
Learning a representation

$$\mathbf{z}_i = U\mathbf{x}_i, \qquad U \in \mathbb{R}^{q \times d}$$

Measure Euclidean distance after the transformation.

$$d_U(\mathbf{x}_i, \mathbf{x}_j) = \|U\mathbf{x}_i - U\mathbf{x}_j\|$$

The transformation can rescale, rotate, and remove directions.



Original space

Horizontal direction compressed

The induced Mahalanobis distance

$$d_U^2(\mathbf{x}_i, \mathbf{x}_j) = \|U(\mathbf{x}_i - \mathbf{x}_j)\|^2$$

$$= (\mathbf{x}_i - \mathbf{x}_j)^\top U^\top U(\mathbf{x}_i - \mathbf{x}_j)$$

$$= (\mathbf{x}_i - \mathbf{x}_j)^\top M(\mathbf{x}_i - \mathbf{x}_j), \qquad M = U^\top U$$

Learning a linear embedding is equivalent to learning this quadratic distance.

Pairwise supervision

$\mathcal{S} = \{(i, j) : \text{the pair should be similar}\}$

$\mathcal{D} = \{(i, j) : \text{the pair should be dissimilar}\}$

Class labels can generate pairs, or the pairs can be supplied directly.

Attraction alone is insufficient: mapping everything to one point would win.

A classical pairwise metric objective

$$\min_{M \succeq 0} \sum_{(i,j) \in \mathcal{S}} d_M^2(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to} \quad \sum_{(i,j) \in \mathcal{D}} d_M(\mathbf{x}_i, \mathbf{x}_j) \geq 1$$

The objective brings similar pairs together.

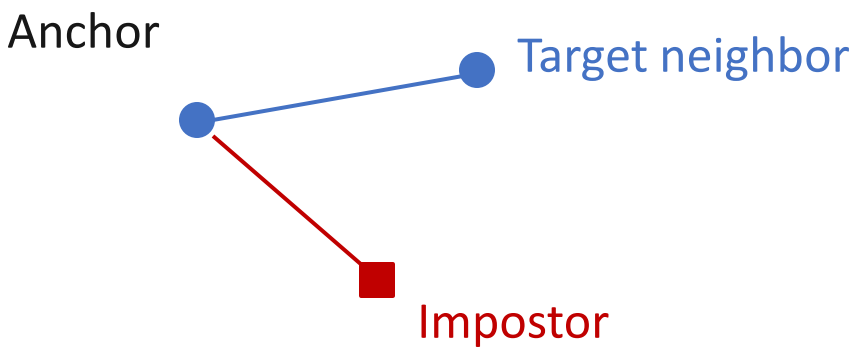
The constraint requires overall separation and excludes the zero matrix.

Large-margin nearest neighbors

Choose nearby same-class observations as target neighbors.

Learn a distance that keeps these targets close.

Penalize a different-class observation that enters their margin.



Pull target closer

Push impostor beyond the target margin

Defining the target-neighbor set

For each training observation, choose its nearest same-class observations.

$\mathcal{T}_i =$ indices of the k nearest same-class observations to $\mathbf{x}_i$

$$j \in \mathcal{T}_i \implies y_j = y_i, \qquad y_l \neq y_i$$

Choose these targets using the initial distance and keep the sets fixed during training.

Writing the nearest-neighbor margin

$j \in \mathcal{T}_i$ is a target, $y_l \neq y_i$ identifies a different-class point.

$$d_M^2(\mathbf{x}_i, \mathbf{x}_l) \geq d_M^2(\mathbf{x}_i, \mathbf{x}_j) + m$$

$$m + d_M^2(\mathbf{x}_i, \mathbf{x}_j) - d_M^2(\mathbf{x}_i, \mathbf{x}_l) \leq 0$$

The different-class point should be farther away by at least the margin.

A penalty for a violated margin

$$\ell_{ijl} = \left[m + d_M^2(\mathbf{x}_i, \mathbf{x}_j) - d_M^2(\mathbf{x}_i, \mathbf{x}_l)\right]_+$$

$$[u]_+ = \max(0, u), \qquad m > 0$$

A satisfied margin gives zero loss. A violation gives a positive penalty.

We minimize this loss to reduce margin violations.

A numerical nearest-neighbor margin

$$j \in \mathcal{T}_i, \quad y_l \neq y_i, \quad m = 1, \quad d_M^2(\mathbf{x}_i, \mathbf{x}_j) = 0.5$$

$$d_M^2(\mathbf{x}_i, \mathbf{x}_l) = 0.8 \quad \Longrightarrow \quad \ell_{ijl} = \max(0, 1 + 0.5 - 0.8) = 0.7$$

The different-class point is farther away, but the gap is only 0.3.

$$d_M^2(\mathbf{x}_i, \mathbf{x}_l) = 1.5 \quad \Longrightarrow \quad \ell_{ijl} = \max(0, 1 + 0.5 - 1.5) = 0$$

The LMNN objective

$$\min_{M \succeq 0} \quad (1-\mu) \sum_i \sum_{j \in \mathcal{T}_i} d_M^2(\mathbf{x}_i, \mathbf{x}_j)$$

$$+\mu \sum_i \sum_{j \in \mathcal{T}_i} \sum_{l: y_l \neq y_i} \ell_{ijl}, \qquad 0 < \mu < 1$$

The first term pulls target neighbors closer.

The second penalizes violations of their separation margin.

From a matrix to a neural network

$$\mathbf{z} = U\mathbf{x} \qquad \longrightarrow \qquad \mathbf{z} = f_U(\mathbf{x})$$

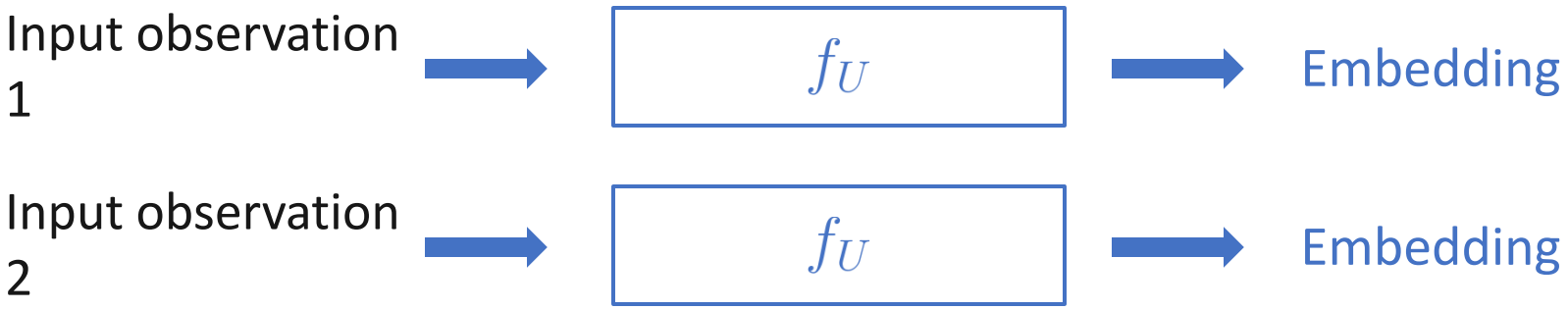
U denotes the trainable weights of the encoder.

$$d_{ij} = \|f_U(\mathbf{x}_i) - f_U(\mathbf{x}_j)\|$$

The network learns a nonlinear mapping from each input to an embedding vector.

A Siamese network uses shared parameters

Two observations pass through the same embedding function.



All branches share the same parameters.

Pair label + distance determine the loss.

Shared encoder, different training losses

A Siamese network applies the same encoder and weights to its inputs.

The loss specifies which relationships the embeddings should learn.

Pairwise contrastive loss uses similar and dissimilar pairs.

Triplet loss compares an anchor with a positive and a negative example.

Why use pairwise contrastive loss?

Suppose the task is to decide whether two photographs show the same person.

Training supplies pairs labeled “same person” or “different people”.

Learn a distance that is small for matches and sufficiently large for non-matches.

At prediction time, compare the distance with a threshold chosen on validation data.

Contrastive loss for a pair

$s_{ij} = 1$ for a similar pair, $s_{ij} = 0$ otherwise

$$\ell_{ij} = s_{ij} d_{ij}^2 + (1 - s_{ij})[m - d_{ij}]_+^2$$

Similar pairs are penalized for being far apart.

Dissimilar pairs are penalized only while they are closer than the margin.

Understanding the two loss curves



The positive-pair loss grows with distance.

The negative-pair loss becomes zero once the margin is reached.

The direction of the contrastive update

Similar pair: $\frac{\partial \ell}{\partial d} = 2d$

Dissimilar pair, $0 < d < m$: $\frac{\partial \ell}{\partial d} = -2(m - d)$

Dissimilar pair, $d \geq m$: $\frac{\partial \ell}{\partial d} = 0$

Gradient descent decreases the first distance and increases the second.

Why use triplet loss?

Suppose a query photograph is used to retrieve photographs of the same person.

For each query, a matching photograph should rank ahead of a non-matching one.

A triplet directly compares these two distances to the same query.

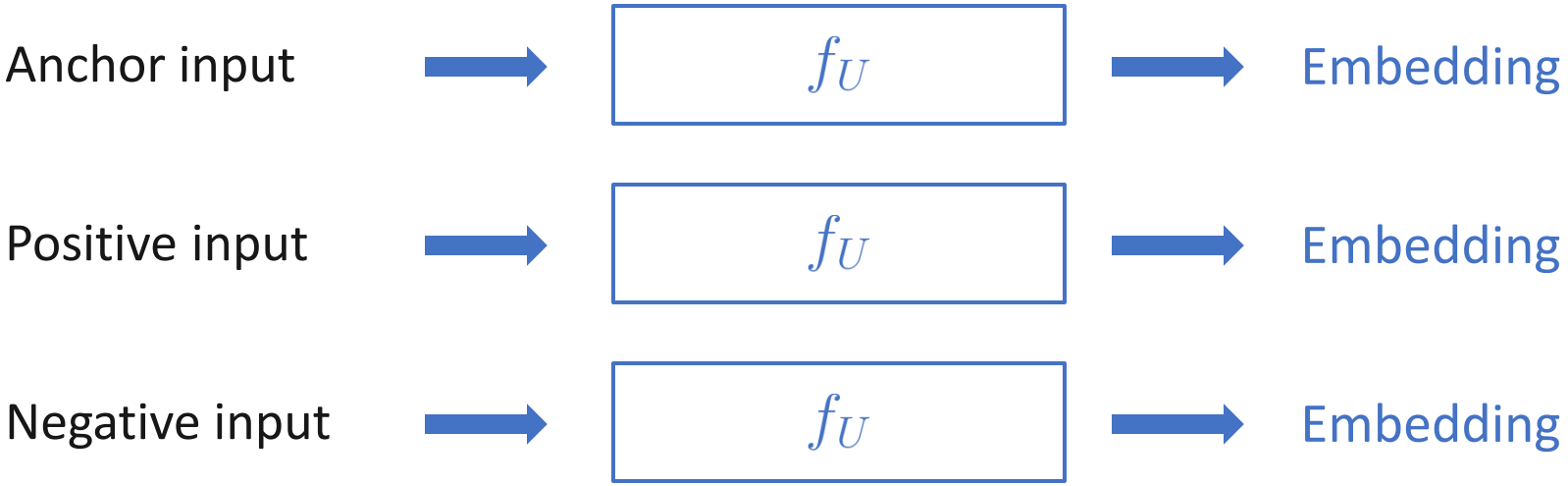
The margin asks for a clear separation between the match and the non-match.

Triplets express relative similarity

Anchor: the reference observation.

Positive: an observation that should be close to the anchor.

Negative: an observation that should be farther away.



All branches share the same parameters.

A triplet example before the formula

An anchor photograph is compared with a matching and a non-matching photograph.

Squared distance to the positive $= 2,$ $m = 1$

Required squared distance to the negative $\geq 2 + 1 = 3$

A squared distance of 2.1 to the negative leaves a margin shortfall of 0.9.

Deriving the triplet loss

$$\mathbf{z}_a = f_U(\mathbf{x}_a), \quad \mathbf{z}_p = f_U(\mathbf{x}_p), \quad \mathbf{z}_n = f_U(\mathbf{x}_n)$$

$$\|\mathbf{z}_a - \mathbf{z}_n\|^2 \geq \|\mathbf{z}_a - \mathbf{z}_p\|^2 + m$$

$$\|\mathbf{z}_a - \mathbf{z}_p\|^2 - \|\mathbf{z}_a - \mathbf{z}_n\|^2 + m \leq 0$$

$$\ell_{apn} = \left[\|\mathbf{z}_a - \mathbf{z}_p\|^2 - \|\mathbf{z}_a - \mathbf{z}_n\|^2 + m\right]_+$$

Training minimizes the triplet loss

$$\ell_{apn} = \max\left(0,\ m + \|\mathbf{z}_a - \mathbf{z}_p\|^2 - \|\mathbf{z}_a - \mathbf{z}_n\|^2\right)$$

The hinge measures how far the distance gap falls short of the margin.

$$\min_U \frac{1}{|\mathcal{Q}|} \sum_{(a,p,n)\in\mathcal{Q}} \ell_{apn}, \qquad \mathcal{Q} = \text{set of training triplets}$$

Update the shared encoder weights to reduce the average loss over training triplets.

A numerical triplet example

$$m = 1, \qquad d_{ap}^2 = 0.5, \qquad d_{an}^2 = 0.8$$

$$\ell = [0.5 - 0.8 + 1]_+ = 0.7$$

The negative is farther away, but the margin is not yet satisfied.

$$d_{an}^2 = 1.8 \quad \Longrightarrow \quad \ell = [0.5 - 1.8 + 1]_+ = 0$$

Which way do the embeddings move?

For an active triplet, differentiate the expression inside the hinge.

$$\nabla_{\mathbf{z}_a} \ell = 2(\mathbf{z}_a - \mathbf{z}_p) - 2(\mathbf{z}_a - \mathbf{z}_n) = 2(\mathbf{z}_n - \mathbf{z}_p)$$

$$\nabla_{\mathbf{z}_p} \ell = 2(\mathbf{z}_p - \mathbf{z}_a), \qquad \nabla_{\mathbf{z}_n} \ell = 2(\mathbf{z}_a - \mathbf{z}_n)$$

A descent step pulls the positive toward the anchor and pushes the negative away.

Informative pairs and triplets

Easy triplets already satisfy the margin and contribute zero loss.

Choose useful violations, while avoiding a diet of mislabeled outliers.

$$\widetilde{\mathbf{z}} = \frac{\mathbf{z}}{\|\mathbf{z}\|}$$

Unit normalization fixes scale, so margins must suit the distance range.

Supervised contrastive learning

One anchor can have several positives in the same training batch.

Compare it with multiple positives and negatives together.

The loss encourages same-class embeddings to have high relative similarity.

This extends the pair-and-triplet viewpoint to a batch-level objective.

Classification with the learned embedding

1. Learn the embedding using only training observations and their relationships.

2. Store the embeddings and labels of the training observations.

3. Embed a new query and find its nearest stored embeddings.

4. Predict by a neighbor vote, or return the closest observations for retrieval.

Three methods, three assumptions

Naive Bayes: features are conditionally independent within each class.

Nearest neighbors: nearby observations tend to share a label.

Metric learning: supervision can make the neighborhoods more useful.

The final classifier can remain simple even when its representation is learned.