A guide to notation used in the course. Let me know if things are missing from this document that are not obvious.

Part 1: Supervised Learning

Throughout the course, we use $n$ as the number of training examples and $d$ as the number of the features. We use $i$ when indexing a training example and $j$ when indexing a feature. We use $x_{ij}$ as feature $j$ in training example $i$, and we use $y_i$ as the label of example $i$ (so there are $n$ values $y_1, y_2, \ldots, y_n$). We use $y$ as a list of the $n$ class labels, containing the label $y_i$ in position $i$. We use $x_i$ as the list of all features for example $i$, so $x_i$ has $d$ elements $x_{i1}, x_{i2}, \ldots, x_{id}$ (and there are $n$ lists $x_1, x_2, \ldots, x_n$). We use $X$ as an $n \times d$ matrix containing all the features, so $x_{ij}$ is element $(i, j)$ of $X$ and $x_i$ gives the elements of row $i$ of $X$. We use $x^j$ to refer to all elements of column $j$, which is the list of values of feature $j$ across all the $n$ training examples.

Throughout the course, we use $t$ as the number of test examples, and $\tilde{X}$ refers to a $t \times d$ matrix containing the test features. The notation $\tilde{x}_i$ refers to the features of test example $i$, while $\tilde{x}_{ij}$ refers to feature $j$ in test example $i$. We use $\tilde{y}$ as the true labels of the test examples, and $\tilde{y}_i$ as the label of test example $i$. We use $\hat{y}_i$ as the prediction of a model on example $i$, whether the prediction is made on training data or validation or test data (it should be obvious or not relevant from context).

When discussing validation sets, $X_{\text{train}}$ and $y_{\text{train}}$ are used as the subsets that we train on, while $X_{\text{validate}}$ and $y_{\text{validate}}$ are used as the subsets that we validate on. We use $E$ to denote a generic prediction error, and usually this is followed with a subscript. For example, $E_{\text{train}}$ is the training error, $E_{\text{test}}$ is the test error, $E_{\text{approx}}$ is the approximation error.

We use $c$ as a class label, and occasionally use $n_c$ as the number of training examples in class $c$. We use the letter $k$ generically throughout the course as something we count, and $\epsilon$ as a generic number that we want to be small.

Some method-specific notations used in this section:

- We use $t$ as a particular decision stump threshold, and $k$ as the number of thresholds.
- $p(y_i = \text{"spam"}|x_i)$ is used for the probability that the label $y_i$ takes the value “spam” given that the features are $x_i$.
- $p(y_i|x_i)$ is used for the probability that the label is $y_i$ given that the features are $x_i$ (for example, $y_i$ could be “spam” or “not spam” but without specifying a particular value).
- In the naive Bayes section, we’re a little sloppy in that we use the same notation for the MLE on the training data and the true population value.

---

1When talking about two training examples, we sometimes use $j$ as the index of the second training example.
• Naive Bayes uses $n_{cjk}$ as the number of times that feature $j$ is equal to $k$ and the class label is $c$.
• Naive Bayes uses $p(\text{hello}|\text{spam})$ as short for $p(x_{ij} = \text{“hello”} \mid y_i = \text{“spam”})$.
• Decision theory slides use $\text{cost}(\hat{y}_i, y_i)$ as the cost of prediction $\hat{y}_i$ when the true label is $y_i$.
• In the norm slides we use $r$ as a generic vector.

Part 2: Unsupervised Learning

We use $\hat{y}_i$ as the cluster predicted for example $i$ and $\mathcal{Y}$ as the set of predicted clusters for all $n$ training examples. We use $C$ as the set of incidences of examples assigned to cluster $c$.

We use $W$ as a $k \times d$ matrix where row $c$ contains mean $c$. We use $w_c$ as mean $c$, $w_{\bar{y}_i}$ to refer to the mean of the cluster of example $i$, and $w_{cj}$ to refer to feature $j$ in mean $c$. We use $w^j$ as column $j$ of the matrix $W$. We use $X$ as predicted values of the matrix $X$, and similarly $\hat{x}_i$ are predicted values of $x_i$ and $\hat{x}_{ij}$ are predicted values of $x_{ij}$.

We use $\mu$ as the mean of the data (with $\mu_j$ being the mean for feature $j$ if we have more than one feature) and $\sigma$ as the standard deviation (with $\sigma_j$ being the standard deviation for feature $j$ if we have more than one feature).

Part 3: Linear models

In this section we start treating $x_i$ and $y_i$ as vectors, so we now have to be careful about whether vectors are row-vectors or column-vectors. Our default choice is that everything is a column-vector, so each $x_i$ is a $d \times 1$ vector and $y$ is an $n \times 1$ vector. Since $x_i$ is now a column-vector, we need to be careful to define row $i$ of $X$ as $x_i^T$ (instead of just $x_i$).

We use $w$ as the $d \times 1$ vector of regression weights. We normally index into $w$ using $w_j$. We sometimes add a y-intercept (“bias”) variable and use $w_0$ to denote this variable (in some settings later in the course $\beta$ is used instead of $w_0$).

We use $\nabla f(w)$ to denote the gradient of a function $f$ with respect to $w$. Assuming $w$ has length $d$, this is a $d \times 1$ vector where position $j$ contains the partial derivative of $f$ with respect to $w_j$. We use $r$ as the vector of “residuals”, $r = Xw - y$. An individual element $i$ of $r$ would be $r_i = w^T x_i - y_i$.

Gradient descent uses $w^t$ as the parameter vector on iteration $t$ (so $t$ has a separate meaning than “number of test examples” here). The distinction between $w^t$ (iteration $t$ of gradient descent) and $w^j$ (column $j$ of matrix $W$) should be clear from the context. We use $\alpha^t$ as the step size on iteration $t$. We use $w^*$ as a minimum of $f(w)$. Stochastic gradient uses $f_i$ to refer to the loss function on example $i$.

We use $Z$ as an $n \times k$ matrix of features obtained under a change of basis, and $z_i$ as the list of $k$ features in the new basis for example $i$. When we do linear regression under a change of basis, we use $v$ as the $k \times 1$ vector of parameters (instead of the usual $d \times 1$ vector $w$). We use $\tilde{Z}$ as the transformation of test data $\tilde{X}$.

We use $\lambda$ as the (scalar) regularization parameter. It is assumed to be non-negative (and will almost always be positive).

We use $\text{sign}(\alpha)$ as a function that return +1 if $\alpha$ is positive and −1 if $\alpha$ is negative.

Multi-class classification uses the same matrix $W$ as we used for k-means, and we use $w_{y_i}$ as the $w_c$ value for the true label $y_i$.

We use $h(z_i)$ as the sigmoid function applied element-wise to a vector $z_i$.

Some method-specific notation used in this section:
• $p$ is used as the degree of the polynomial in the polynomial basis, and we sometimes use $Z_p$ when we want to specify specifically that we’ve used a degree-$p$ basis.

• We use $K$ as the $n \times n$ Gram matrix, containing $z_i^T z_j$ in position $(i,j)$. We use $\hat{K}$ as the $t \times n$ matrix containing $\hat{z}_i^T z_j$ in position $(i,j)$. We use $u$ as the $n \times 1$ parameter vector when doing kernel methods for linear models. The kernel function is written as $k(x_i, x_j)$.

• When introducing MLE/MAP, we use $D$ as generic data (indexed by $D_i$ if it splits into IID training examples), $w$ as generic parameters, and $\hat{w}$ as the predicted MLE or MAP value of $w$.

**Part 4: Latent-Factor Models**

Linear latent-factor models use the approximation $X \approx ZW$, where we use the same notation for $Z$ and $W$ as above: $Z$ is $n \times k$ with $z_i^T$ as the rows and $z_{ic}$ as individual elements, $W$ is $k \times d$ with $w_j^T$ as the rows and $w_j$ as the columns and $w_{cj}$ as the individual elements. To avoid expressions like $(w_j)^T z_i$, for inner products in this section we sometimes use notation $(w, x)$ to represent the inner product $w^T x$.

**Part 5: Neural Networks**

This section continues using the same notation, but we now use $W^{(l)}$ and $Z^{(l)}$ as the values in layer $l$. We also use $w_{c0}$ as the bias on hidden unit $c$, and $m$ as the number of layers.

When we introduce convolutions we use $x$ as signal, $w$ as a filter, and $z$ as the output of the filter.