## Dados de dígitos manuscritos

No conjunto de dados de dígitos manuscritos, temos 60000 dígitos escritos a mão em imagens de dimensão $28 \times 28$. O conjunto de dados foi separado em conjunto de treino e validação, onde 25% dos dados foi utilizado como conjunto de validação, ou seja, uma amostra de 45000 dígitos será utilizada para construir os modelos, e os dígitos restantes serão utilizados para avaliar a qualidade de ajuste dos modelos.

Em seguida os pixels foram normalizados para a escala $[0, 1]$, (equivalente a por a imagem na escala de cinza) para que alguns dos modelos utilizados convirja mais rápido.

Com os dados normalizados e separados em conjunto de treino e teste, serão construídos três modelos para realizar as predições das imagens:

- Lasso Multinomial
- Boosting
- Rede Neural Convolucional

### Modelo linear com regularização

Como o problema é diferenciar 10 dígitos utilizando 784 pixels, isso indica que o problema é de classificação com mais de duas classes. Logo, será ajustado um modelo multinomial. E como diversos pixels geralmente não são preenchidos para escrever os dígitos, é interessante realizar selecionar os pixels importantes para a classificação de cada dígito. Por isso, será ajustado um modelo Lasso Multinomial. Para a escolha do parâmetro de penalização foi gerado 100 hiper-parâmetros de penalização, e foi escolhido o melhor hiper-parâmetro através de validação cruzada utilizando apenas o conjunto de treino.

Na figura 1 é apresentado os pixels que não foram nulos para a classificação de cada dígito, com o dígito referente em verde no canto superior esquerdo de cada imagem.

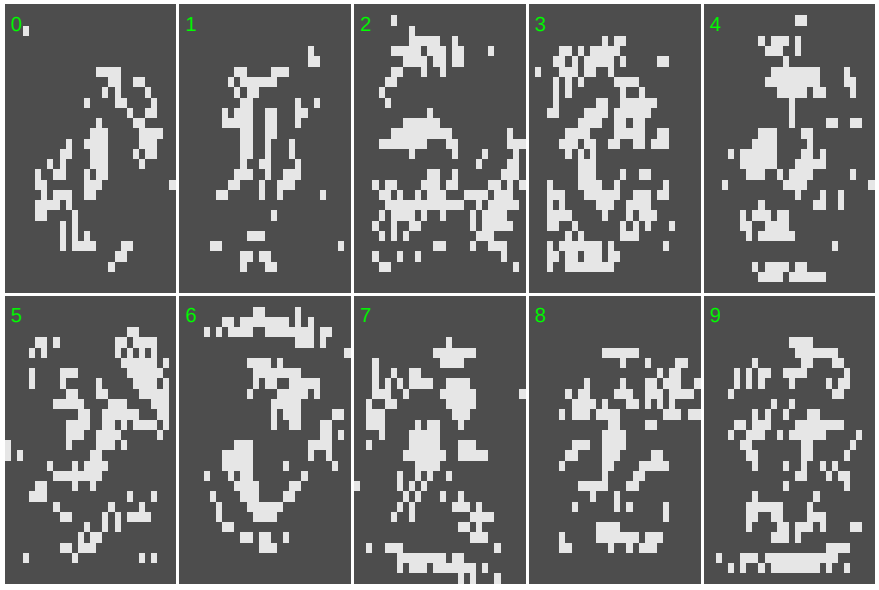


Figura 1: Pixels não nulos para cada dígito pelo modelo lasso.

Pela Figura 1, é possível ver os principais pixels para a classificação de cada dígito. Pode-se notar que a partir dos pixels não nulos, é possível desenhar o dígito que deseja-se classificar.

### Modelo baseado em árvore

Existe vários tipos possíveis de modelos baseados em árvore que podem ser utilizados. Porém, para casos de classificação de imagem, mais especificamente a classificação de dígitos do MNIST, modelos de *boosting* conseguem oferecer um bom ajuste. Por conta disso, será ajustado um modelo de boosting utilizando o pacote XGBoost.

Para a seleção dos hiper-parâmetros, foi ajustado 75 modelos para selecionar a taxa de aprendizagem (`eta`), a profundidade máxima da árvore (`max_depth`) e a perda mínima para criar outra partição da árvore (`min_child_weight`). Os parâmetros escolhidos foram:

- eta: 0.1
- max_depth: 6
- min_child_weight: 4

O número de árvores foi controlado pelo conjunto de validação, porém foi definido que o número máximo de árvores seria 500.

```
## [1]  train-mlogloss:1.958892 eval-mlogloss:1.968458
## Multiple eval metrics are present. Will use eval_mlogloss for early stopping.
## Will train until eval_mlogloss hasn't improved in 5 rounds.
##
## [11] train-mlogloss:0.785277 eval-mlogloss:0.829439
## [21] train-mlogloss:0.415197 eval-mlogloss:0.472737
## [31] train-mlogloss:0.248233 eval-mlogloss:0.313767
## [41] train-mlogloss:0.163747 eval-mlogloss:0.234302
## [51] train-mlogloss:0.115883 eval-mlogloss:0.189373
## [61] train-mlogloss:0.086443 eval-mlogloss:0.161798
## [71] train-mlogloss:0.066508 eval-mlogloss:0.143448
## [81] train-mlogloss:0.052754 eval-mlogloss:0.130958
## [91] train-mlogloss:0.042125 eval-mlogloss:0.121397
## [101]    train-mlogloss:0.034435 eval-mlogloss:0.114478
## [111]    train-mlogloss:0.028727 eval-mlogloss:0.109501
## [121]    train-mlogloss:0.024053 eval-mlogloss:0.105037
## [131]    train-mlogloss:0.020432 eval-mlogloss:0.101447
## [141]    train-mlogloss:0.017443 eval-mlogloss:0.098328
## [151]    train-mlogloss:0.015068 eval-mlogloss:0.095521
## [161]    train-mlogloss:0.013146 eval-mlogloss:0.093366
## [171]    train-mlogloss:0.011537 eval-mlogloss:0.091504
## [181]    train-mlogloss:0.010180 eval-mlogloss:0.089845
## [191]    train-mlogloss:0.009066 eval-mlogloss:0.088509
## [201]    train-mlogloss:0.008145 eval-mlogloss:0.087516
## [211]    train-mlogloss:0.007360 eval-mlogloss:0.086497
## [221]    train-mlogloss:0.006712 eval-mlogloss:0.085739
## [231]    train-mlogloss:0.006111 eval-mlogloss:0.084994
## [241]    train-mlogloss:0.005596 eval-mlogloss:0.084329
## [251]    train-mlogloss:0.005183 eval-mlogloss:0.083704
## [261]    train-mlogloss:0.004804 eval-mlogloss:0.083201
## [271]    train-mlogloss:0.004493 eval-mlogloss:0.082794
## [281]    train-mlogloss:0.004191 eval-mlogloss:0.082423
## [291]    train-mlogloss:0.003931 eval-mlogloss:0.082101
## [301]    train-mlogloss:0.003706 eval-mlogloss:0.081813
## [311]    train-mlogloss:0.003501 eval-mlogloss:0.081561
## [321]    train-mlogloss:0.003322 eval-mlogloss:0.081354
```

```
## [331]    train-mlogloss:0.003155 eval-mlogloss:0.081104
## [341]    train-mlogloss:0.003011 eval-mlogloss:0.080882
## [351]    train-mlogloss:0.002876 eval-mlogloss:0.080647
## [361]    train-mlogloss:0.002757 eval-mlogloss:0.080391
## [371]    train-mlogloss:0.002641 eval-mlogloss:0.080302
## [381]    train-mlogloss:0.002535 eval-mlogloss:0.080205
## [391]    train-mlogloss:0.002441 eval-mlogloss:0.080048
## [401]    train-mlogloss:0.002355 eval-mlogloss:0.080015
## [411]    train-mlogloss:0.002275 eval-mlogloss:0.079870
## [421]    train-mlogloss:0.002201 eval-mlogloss:0.079746
## [431]    train-mlogloss:0.002136 eval-mlogloss:0.079710
## Stopping. Best iteration:
## [428]    train-mlogloss:0.002155 eval-mlogloss:0.079702
```

**Redes neurais**

Para o modelo de rede neural, foi considerado uma rede neural convolucional base com acurácia 0.9897. Para a rede base, foi utilizada uma camada convolucional e uma camada escondida, todas camadas tiveram função de ativação `relu`. O otimizador utilizado foi o `Adam` com taxa de aprendizagem 0,0001.

Em seguida foi ajustado diversas redes, manipulando a quantidade de camadas intermediárias, adicionando camada de `dropout` e manipulando o otimizador (Ruder apresentou diversos otimizadores baseados em gradiente descendente). A estrutura da rede final é:

- camada convolucional com 32 filtros, janela deslizante $5 \times 5$ e ativação relu;
- camada de `max_pooling` de dimensão $2 \times 2$;
- camada de `dropout` com taxa 0.2;
- camada convolucional com 64 filtros, janela deslizante $5 \times 5$ e ativação relu;
- camada de `max_pooling` de dimensão $2 \times 2$;
- camada de `dropout` com taxa 0.2;
- camada de achatamento (`flatten`);
- camada `dense` com 512 neurônios e ativação relu;
- camada de `dropout` com taxa 0.2;
- a última camada é uma `dense` com ativação softmax de 10 neurônios para classificar o dígito.

O otimizador escolhido foi o `nadam` que é similar ao `adam` mas contém momento para acelerar a convergência. Foi definido que o ajuste seria parado se houvesse 5 épocas sem melhora, e o modelo foi ajustado em lotes de tamanho 64.

## Comparação dos modelos

Para a comparação dos modelos será utilizado a acurácia. Na Tabela 1 é apresentado a acurácia para o conjunto de treino (Acurácia Dentro) e validação (Acurácia Fora).

Tabela 1: Acurácias dos modelos ajustados.

| Modelo | Acurácia Dentro | Acurácia Fora |
|---|---|---|
| Multinomial Lasso | 0.9200667 | 0.9089333 |
| Boosting | 1.0000000 | 0.9748000 |
| Rede Neural Convolucional | 0.9968444 | 0.9903333 |

Podemos ver pela Tabela 1 que o modelo de rede neural foi o que teve as melhores predições no conjunto de validação. Nota-se também que o modelo de Boosting predizeu perfeitamente no conjunto de treino, o que mostra que esse modelo conseguiu decorar o conjunto de treino. Porém, conseguiu predizer corretamente na maior parte no conjunto de validação.

Como o modelo de rede neural convolucional foi o que teve a maior acurácia no conjunto de validação, ele será o modelo escolhido para realizar as predições no conjunto de teste. Para apresentar um pouco das predições do modelo escolhida, na Figura 2 é apresentado a matriz de confusão do modelo no conjunto de validação.

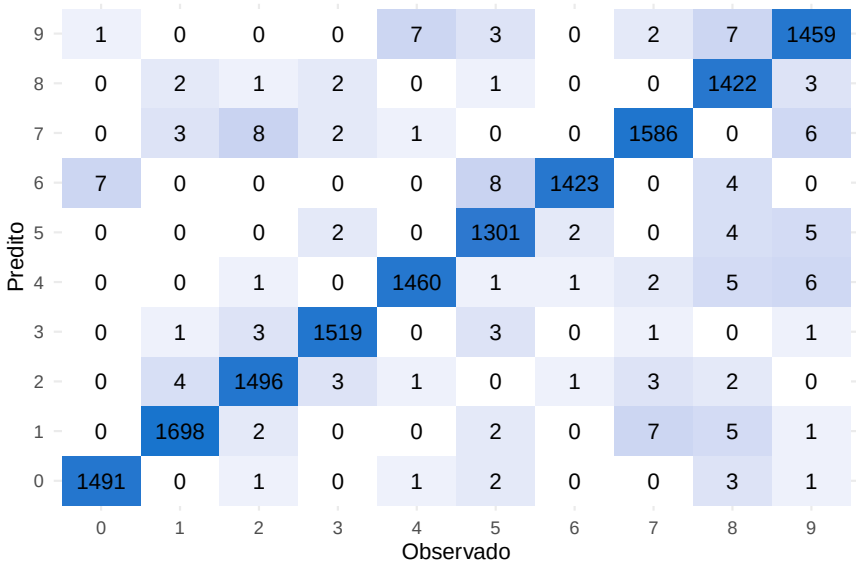


Figura 2: Matriz de confusão para o modelo de rede neural convolucional para o conjunto de dados de validação.

Pela Figura 2, não se percebe um dígito em que o modelo apresentou dificuldades de prever.

Na Figura 3 se encontram algumas predições no conjunto de validação. As predições corretas estão na cor verde, e as incorretas estão na cor vermelha com a predição correta dentro dos parênteses.

Pela Figura 3, nota-se alguns dígitos que são facilmente reconhecíveis, porém o modelo predizeu incorretamente. Mas como não se espera que um modelo consiga acertar em todo o conjunto de validação, e como ele teve uma ótima acurácia, ele será o modelo escolhido para realizar as predições no conjunto de teste.

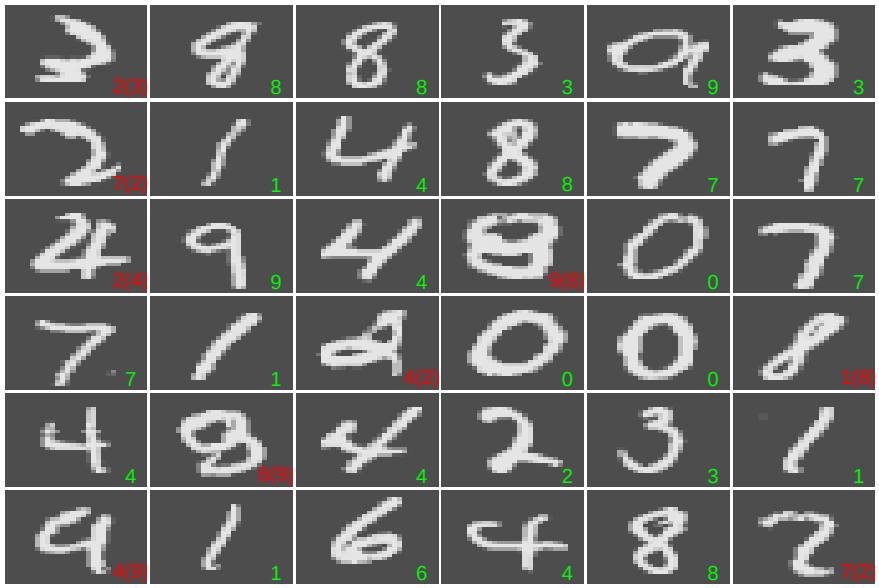


Figura 3: Amostra das classificações da rede neural para o conjunto de validação.